Higher Order Logic

Daniel Leivant

Contents

1 Introduction

Higher order logics, long considered by many to be an esoteric subject, are increasingly recognized for their foundational importance and practical usefulness, notably in Theoretical Computer Science. In this chapter we try to present a survey of some issues and results, without any pretense of completeness. Our choice of topics is driven by an attempt to cover the foundational aspects of higher order logic, and also to briefly point to some areas of current and potential applications.

The chapter falls into two parts. The first part, consisting of sections 1 through 4, is designed to bring forth the essential issues and facts of the topic. Section 1 is intended to motivate our interest in higher order logic, by presenting selected examples of the expressive power of second order logic, and contrasting them with the limitations of first order logic. In section 2 we define more precisely the canonical (intended) semantics of second and higher order logics. This leads us to section 3, where the proof theory of second order logics is considered. Section 4 discusses issues touching on the question of whether second and higher order logics are genuine logics. Starting with a sample of major differences between first order and higher order logic, we consider the foundational merits of second order logic, and discuss the view of higher order logic as mathematical theories rather than genuine logics.

The second part of this chapter surveys slightly more technical concepts and issues. We mostly describe issues in very broad lines, with pointers to the literature, only occasionally explaining a concept or providing a proof outline for an important result. Section 5 presents some useful options for restricting second order logic: restrictions of the language (monadic second order logic and fixpoint logics), of the semantics (weak second order

logic) and of the proof theory (predicative higher order logic, i.e. the ramified theory of types). Section 6 discusses the role of higher order logic in mathematical practice: we survey some relations between higher order logic and two branches of mathematics, set theory and mathematical analysis. We then discuss two general issues related to higher order formalization of mathematics: second order axioms versus first order schemas, and the effect of higher order reasoning on shortening proofs. The seventh and last section briefly describe some relations between higher order logic and programming and computing: the use of higher order data in programming, the computational nature of natural deduction proofs, the role of higher order logic in the meta-theory of formal systems, and the use of higher order logic to provide machine-independent characterizations of computational complexity classes.

Given the space and subject constraints of a handbook chapter, many important aspects of higher order logic receive here a scant treatment, and others are omitted altogether, notably most uses of higher order constructs in mathematical practice, in recursion theory, and in computer science. Such choices of topics can not be independent of an author's interests and background. My hope is, though, that the chapter touches on the central issues of the field, and that it offers some useful organizing principles to a broad and diffuse subject. The interested reader will find a large and growing number of textbooks, monographs, and surveys about higher order logic and related issues, of which several are pointed to in this text.

2 The expressive power of second order Logic

2.1 The language of second order logic

The language of (full) second order logic is simply the language of first order logic augmented with *second order variables*, that is, variables ranging over relations and functions (of all arities). Given a vocabulary[1] V, V-terms and atomic V-formulas are defined as in first order logic (with equality), but using also function-variables and relation-variables in complete analogy to function-constants and relation-constants. Compound V-formulas are then generated from atomic formulas using the usual propositional connectives as well as quantifiers over all variables, including the function-variables and relation-variables. A formula with no free variable is a *closed* formula, or a *sentence*.

[1] The notion of a vocabulary (= signature, symbol-set) is the same as in first order logic, i.e. a vocabulary V consists of disjoint sets of identifiers for element-constants, function-constants and relation-constants, and a function $arity_V$ that assigns a non-negative integer to each function-constant and relation-constant. We follow the traditional use of lower-case letters from the end of the Roman alphabet for element-variables, $f, g, h \ldots$ for function-variables, and upper-case Roman letters for relation-variables. Whenever convenient we superscript identifiers with their arity.

The standard semantics of second order V-formulas is defined with respect to usual V-structures (of first order logic). The truth of V-formulas in a V-structure is straightforward: function-variables of arity r range over (total) functions of r arguments over the structure's universe, and relation-variables of arity r range over r-ary relations.[2] For example, the equivalence

$$(x = y) \;\leftrightarrow\; \forall R\ R(x) \leftrightarrow R(y)$$

is true in all structures, and may be used as a definition of the equality predicate.

Quantification over relations (and/or functions) greatly enhances the expressive power of first order formulas. In the next few subsections we consider a few significant examples of that extended expressiveness. Let us agree to use the phrase *theory* for a set of formulas, and *V-theory* for a set of V-formulas. A *first order theory* is then simply a set of first order formulas, and a *second order theory* a set of second order formulas.

A structure $\mathcal{S}$ is a *model* of a theory Γ if every formula in Γ is true in $\mathcal{S}$. A collection of structures is *defined* by a theory Γ if it consists exactly of the models of Γ. If a collection of structures is defined by a first order (resp. second order) theory, then it is *first order* (resp. *second order*) *definable*.

Many limitations of first order logic follow easily from one of its fundamental properties, the Compactness Theorem:

Theorem 2.1.1. [Compactness] *Suppose Γ is a first order V-theory, such that every finite subtheory of Γ has a model. Then Γ has a model (which is countable if V is countable).*

2.2 Expressing size

Consider first a topic that comes up as one of the first exercises in logic textbooks: stating the size of a structure and of subsets of its universe.

Theorem 2.2.1. *The collection of finite structures is not first order definable, but it is definable already by a single second order sentence.*

Proof. 1. Assume Γ is a first order theory that defines the finite structures, and let Γ' be Γ augmented with all sentences $\sigma_{\geq 1}$, $\sigma_{\geq 2}$, ..., where $\sigma_{\geq k}$ states that there exist k pairwise distinct elements.[3] Then every finite subtheory of Γ' has a model, and so Γ' has some model, $\mathcal{S}$, by the Compactness Theorem. Since $\mathcal{S}$ is a model of each $\sigma_{\geq k}$, it is infinite. Since $\Gamma \subseteq \Gamma'$, $\mathcal{S}$ is an infinite model of Γ, contradicting the assumption.

---

[2] A more formal definition is given in Section 2.

[3] That is, $\sigma_{\geq n} \;\equiv_{\mathrm{df}}\; \exists x_1 \ldots x_n\ x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge \cdots \wedge x_{n-1} \neq x_n$, with $n(n-1)/2$ conjuncts.

2. A statement that holds exactly of the finite structures is 'every injection is a surjection', i.e.,

$$\sigma_{<\aleph_o} \equiv_{\text{df}} \forall f \quad (\text{Inj}(f) \to \text{Surj}(f))$$

where $\text{Inj}(f) \equiv_{\text{df}} \forall x, y\ (f(x) = f(y) \to x = y)$ and $\text{Surj}(f) \equiv_{\text{df}} \forall u \exists v\ f(v) = u$.

∎

The formula $\sigma_{<\aleph_o}$ can be modified to state that a set (i.e. the extension of a unary relation) A is finite[4]:

$$\sigma_{<\aleph_o}[A] \equiv_{\text{df}} \quad \forall R\ (\ (\forall x, y, z\ R(x,y) \wedge R(y,z) \to R(x,z)) \\ \wedge\ (\forall x \in A)(\exists y \in A) R(x,y)\ \to\ (\exists x \in A)\, R(x,x)\ )$$

Second order formulas can be used to express a great deal about cardinalities. Let

$$B \leq A \quad \equiv_{\text{df}} \quad \exists f\ (\forall y \in B)\, (\exists x \in A)\, f(x) = y.$$

Then $B \leq A$ holds exactly when there is a map from A onto B, i.e. iff $\| B \| \leq \| A \|$.[5] Define, by recurrence on natural numbers n,

$$\sigma_{<\aleph_{n+1}}[A] \equiv_{\text{df}} \quad (\forall X \subseteq A)\ (\ \sigma_{<\aleph_n}[X] \vee (A \leq X)\ ).$$

Then $\sigma_{<\aleph_n}[A]$ is true exactly when $\| A \| < \aleph_n$, and so $\sigma_{<\aleph_{n+1}}[A] \wedge \neg\sigma_{<\aleph_n}[A]$ is true exactly when $\| A \| = \aleph_n$.[6] Similarly, we can state that the universe in hand is of size $< \aleph_{n+1}$:

$$\sigma_{<\aleph_{n+1}} \equiv_{\text{df}} \quad (\forall X)\ (\ \sigma_{<\aleph_n}[X] \vee (U \leq X)\ ),$$

where $U \leq X \equiv_{\text{df}} \exists f\ (\forall y \in X)\, (\exists x)\, f(x) = y.$

---

[4] If R is a binary relation, then we use $(\forall x R y)\varphi$ as an abbreviation for $\forall x\ (\ xRy \to \varphi\ )$, and $(\exists x R y)\varphi$ for $\exists x\ (\ xRy \wedge \varphi\ )$. Also, we write $x \in A$ for $A(x)$, and $X \subseteq A$ for $\forall u\ X(u) \to A(u)$.

[5] $\| S \|$ denotes the cardinality of the set S.

[6] The progression of the $\beth$'s cardinals (by the power-set operation) is also readily expressed. Let

$$(A \leq \mathcal{P}(B)) \quad \equiv_{\text{df}} \quad (\exists R \subseteq A \times B)\, (\forall x, y \in A) \\ (\ x \neq y \to \exists u\, (R(x,u) \wedge \neg R(y,u)) \vee (R(y,u) \wedge \neg R(x,u))\ )$$

$$(\mathcal{P}(B) \leq A) \quad \equiv_{\text{df}} \quad (\exists R \subseteq A \times B)(\forall C \subseteq A)\, (\exists x \in A) \forall u (C(u) \leftrightarrow R(x,u))$$

$(A \leq \mathcal{P}(B))$ states that the mapping $f_R : A \to \mathcal{P}(B)$, given by $f_R(x) = \{u \mid R(x,u)\}$ is an injection, and $(\mathcal{P}(A) \leq B)$ states that f_R is a surjection. From these formulas it is easy to build formulas that capture the sequence $\beth_k$ for $k < \omega$.

2.3 Defining data types

Let V_s be the vocabulary consisting of a single element-constant $\mathbf{0}$ and a single unary function-constant $\mathbf{s}$. Let $\mathcal{N}_s$ be the V_s-structure consisting of the natural numbers with zero and successor as the interpretations of $\mathbf{0}$ and $\mathbf{s}$. By Skolem–Löwenheim's Theorem for first order logic, there is no first order theory that characterizes $\mathcal{N}_s$ up to isomorphism, since any such theory would have non-countable models. What about characterizing $\mathcal{N}_s$ among the countable structures[7]

Theorem 2.3.1.

1. *There is no first order V_s-theory whose only countable model (up to isomorphism) is $\mathcal{N}_s$.*

2. *There is a second order V_s-sentence whose only model (of any cardinality) is $\mathcal{N}_s$.*

Proof. 1. Suppose Γ were a theory as above. Let Γ' be Γ augmented with all formulas of the form $\mathbf{c} \neq \bar{n}$ $(n = 1, 2, \ldots)$, where $\bar{n}$ is the n'th numeral[8] and where $\mathbf{c}$ is a fresh element-constant. Then every finite subtheory of Γ' has a model, and so Γ' has a countable model $\mathcal{S}$, by the Compactness Theorem. The universe of $\mathcal{S}$ has an element, namely the interpretation of $\mathbf{c}$, which is not obtained by finite applications of $\mathbf{s}$ on $\mathbf{0}$. Therefore $\mathcal{S}$ cannot be isomorphic to $\mathcal{N}_s$. But $\mathcal{S}$ is a model of Γ, a contradiction.

2. Let

$$\varphi_N \quad \equiv_{\mathrm{df}} \quad \varphi_{\mathrm{suc}} \wedge \forall x\, N(x)$$

where

$$\varphi_{\mathrm{suc}} \quad \equiv_{\mathrm{df}} \quad \forall x\, \mathbf{s}(x) \neq \mathbf{0} \wedge \forall x, y\, (\, \mathbf{s}(x) = \mathbf{s}(y) \rightarrow x = y\,)$$

$$N[x] \quad \equiv_{\mathrm{df}} \quad \forall R\, (\, R(\mathbf{0}) \wedge \forall u\, (\, R(u) \rightarrow R(\mathbf{s}(u))\,) \rightarrow R(x)\,).$$

φ_{suc} state that the denotations of numerals are all distinct, whereas $\forall x N(x)$ states that all elements are denotations of numerals.[9] ∎

From the second order characterization of $\mathcal{N}_s$ there follows a second order interpretation of the true formulas of Peano's Arithmetic, PA. Let

---

[7] Theorem 2.3.1 shows that a particular countable structure, that of the natural numbers, is characterized up to isomorphism by its second order theory. A natural question is whether every countable structure is so characterized. [Ajtai, 1979] (reported in [Shapiro, 1991, Section 6.6.4], who attributes the result also to Magidor) shows that this is so for structures defined within the Gödel-constructible universe (i.e. assuming $V = L$), but that the general claim is independent of ZFC.

[8] I.e., $\bar{n} =_{\mathrm{df}} \mathbf{s}^{[n]}(\mathbf{0}) = \mathbf{s}(\mathbf{s}(\cdots \mathbf{s}(\mathbf{0}) \cdots))$ (with n applications).

[9] This second order characterization of the natural numbers goes back to Dedekind.

$D_{+,\times}$ be the conjunction of the closures of the defining equations for addition and multiplication.[10] Then a formula ψ in the language of PA is true in the standard model iff the second order formula $\varphi_N \wedge D_{+,\times} \rightarrow \psi^N$ is valid.[11] Moreover, the same holds for ψ a *second order* formula in the language of PA; that is,

Theorem 2.3.2. *The truth, in the standard model, of second order formulas in the language of PA, is effectively reducible to the validity of second order formulas.*

Just as the natural numbers are second order definable, so is any free algebra (i.e. inductive data-type) $\mathbb{A}$. The algebra $\mathbb{A}$ generated from a given finite set V of constant-identifiers and function-identifiers is characterized by the conjunction χ_V of

1. a statement that the denotations of distinct terms are distinct, i.e. the universal closure of the inequalities

$$\mathbf{f}(x_1 \ldots x_r) \neq \mathbf{g}(y_1 \ldots y_k); \quad \mathbf{f}(x_1 \ldots x_r) \neq \mathbf{c}; \text{ and } \quad \mathbf{c} \neq \mathbf{d},$$

   and of the equalities

$$\mathbf{f}(\vec{y}, x, \vec{z}) = \mathbf{f}(\vec{y}, x', \vec{z}) \ \rightarrow \ x = x',$$

   for all $\mathbf{f}, \mathbf{g}$ distinct function-constants, and $\mathbf{c}, \mathbf{d}$ distinct element-constants of V; and
2. a second order sentence stating that all elements are interpretations of terms:

$$A[x] \ \equiv_{\mathrm{df}} \ \forall R \ (\ Cl_A(R) \rightarrow R(x)\ ),$$

   where $Cl_A(R)$ is the conjunction of all formulas $R(\mathbf{c})$ for $\mathbf{c}$ a constant-identifier of V, and of all universal closures of formulas $R(x_1) \wedge \cdots \wedge R(x_k) \rightarrow R(\mathbf{f}(x_1 \ldots x_k))$ where $\mathbf{f}$ is a function-identifier of V (of arity k).

One free algebra particularly relevant to computing is the algebra $\mathbb{W}$ generated from one element-constant, say ϵ, and two successor functions (i.e. unary function constants), say $\mathbf{0}$ and $\mathbf{1}$. The terms here can be identified with the words over $\{0, 1\}$, e.g. $\mathbf{0}(\mathbf{1}(\mathbf{1}(\epsilon)))$ is identified with 011. The set of terms is characterized by the formula[12]

---

[10] I.e., $D_{+,\times} \ \equiv \ \forall x(x+\mathbf{0}=x) \wedge \forall x,y\,(x+\mathbf{s}y=\mathbf{s}(x+y)) \wedge \forall x\,(x\times\mathbf{0}=\mathbf{0}) \wedge \forall x,y\,(x\times\mathbf{s}y=(x\times y)+x)\,)$.

[11] As customary, if R is a unary relation then φ^R will denote the result of relativizing all quantifiers in the formula φ to R. That is, $(\forall x\psi)^R \ \equiv_{\mathrm{df}} \ (\forall x(R(x) \rightarrow \psi^R)$, and $(\exists x\psi)^R \ \equiv_{\mathrm{df}} \ (\exists x(R(x) \wedge \psi^R)$.

[12] We use the customary abbreviation that uses a chain of $\neq$'s.

$$\begin{aligned}\varphi[z] \equiv_{\mathrm{df}} \quad & \forall x \ \ \epsilon \neq \mathbf{0}x \neq \mathbf{1}x \neq \epsilon \\ & \wedge \forall x\, x' \ ((0x = 0x' \vee 1x = 1x') \ \rightarrow x = x' \ ) \\ & \rightarrow \ \forall R \ (\ R(\epsilon) \wedge \forall z \ (\ R(z) \rightarrow R(\mathbf{0}z) \ \wedge R(\mathbf{1}z) \ ) \ \rightarrow \ R(z) \ ).\end{aligned}$$

2.4 Describing processes

Let $V_G = \{\mathbf{a}, \mathbf{b}, \boldsymbol{\rho}\}$ be the vocabulary of digraphs with a source and a target, i.e. $\mathbf{a}$ and $\mathbf{b}$ are element-constants and $\boldsymbol{\rho}$ is a binary relation-constant.

Theorem 2.4.1. *The collection of digraphs with a path from the source to the target is not first order definable, but it is definable by a single second order V_G-sentence.*

Proof. The non-existence of a first order theory defining these digraphs is, again, a simple application of the Compactness Theorem.

On the other hand, the sentence

$$\forall R \ (\ R(\mathbf{a}) \ \wedge \ \forall x, y \ (\ R(x) \wedge \boldsymbol{\rho}(x, y) \rightarrow R(y) \ ) \ \ \rightarrow R(\mathbf{b}) \ )$$

is true in a digraph $\mathcal{G}$ iff there is a $\boldsymbol{\rho}^{\mathcal{G}}$-path from $\mathbf{a}^{\mathcal{G}}$ to $\mathbf{b}^{\mathcal{G}}$. ∎

The existence of paths for binary relations (i.e. the transitive closure operation) is related to the operational semantics of many computation processes, where a computation is defined as a transition-chain of configurations. The contrast between first order and second order formulas in defining the transitive-closure of a relation is therefore symptomatic of a broader contrast in the possibility of defining program semantics in first order vs. second order logic. Let V be a vocabulary that includes a unary function-constant $\mathbf{f}$. Over every V-structure we have the computable partial function

$$\mathit{fix}_{\mathbf{f}}(x) =_{\mathrm{df}} \text{the first iterate } \mathbf{f}^{[k]}(x) \text{ such that } \mathbf{f}^{[k+1]}(x) = \mathbf{f}^{[k]}(x).$$

An argument similar to the proof of Theorem 2.4.1 shows:

Theorem 2.4.2. *The graph of $\mathit{fix}_{\mathbf{f}}$ is defined, uniformly for all V-structures, by a second order formula, but not by a first order theory.*[13]

For a slightly more general example, let L be the loop **while** $\mathbf{g}(x) = \mathbf{0}$ **do** $x := \mathbf{f}(x)$ **end**. Then the input-output relation determined by L is defined, uniformly for all structures, by the formula

$$M_L[u_0, v_0] \ \equiv \ \forall R \ (\ R(u_0) \ \wedge \ \mathit{Prog}_1[R] \ \ \rightarrow R(v_0) \ ),$$

where

[13] I.e., there is no theory Γ and variables u, v for which the following holds: for every structure $\mathcal{S}$, if F is the interpretation of $\mathit{fix}_{\mathbf{f}}$ in $\mathcal{S}$ (as arising from the interpretation in $\mathcal{S}$ of $\mathbf{f}$), then, for all elements a, b of the universe of $\mathcal{S}$, $b = F(a)$ iff $\mathcal{S}, [a/u, b/v] \models \Gamma$.

$$Prog_1[R] \equiv_{\mathrm{df}} \forall w\,(R(w) \wedge \mathbf{g}(w)=0 \rightarrow R(\mathbf{f}(w))).$$

Here L changes the value of a single variable, hence the formula M_L has only one argument for input (u_0) and one for output (v_0). If Q is the loop **while** $\mathbf{g}(x)=y$ **do** P, and the input-output semantics of P (for program variables x and y) is already defined by a formula $M_P[u_0,u_1,v_0,v_1]$, then the relation determined by Q is defined by

$$M_Q[u_0,u_1,v_0,v_1] \quad \equiv \quad \forall R\ (\ R(u_0,u_1)\ \wedge\ Prog_2[R]\ \rightarrow\ R(v_0,v_1)\ ),$$

where

$$\begin{aligned} Prog_2[R] \quad \equiv_{\mathrm{df}} \quad & \forall w_0,w_1,z_0,z_1\,(R(w_0,w_1) \\ & \wedge \mathbf{g}(w_0)=w_1 \wedge M_P[w_0,w_1,z_0,z_1]\ \rightarrow\ R(z_0,z_1)). \end{aligned}$$

It is easy to exhibit similar definitions for most basic programming constructs, including recursive procedures and modules [Leivant, 1983].

2.5 Expressing convergence using second order validity

The definability of free-algebras by second order formulas can be extended to a characterization of the total computable functions. We summarize the argument of [Leivant, 1983; Leivant, 1994]. Of the many computation calculi that generate the total recursive functions one that lends itself naturally to this model theoretic setting are equational programs; a rudimentary form of such programs would serve us best.[14] Fix a free algebra $\mathbb{A}$. We posit an unlimited supply of r-ary *program-functions* (for each arity $r>0$), which are identifiers distinct from the constructors of $\mathbb{A}$. The *terms* are generated from the constructors of $\mathbb{A}$, free variables, and program-functions. A *statement* over $\mathbb{A}$ is an equation between terms. A set of statements is a *program-body*. A *program* (over $\mathbb{A}$) is a pair, $(P,\mathbf{f})$, where P is a program-body, and $\mathbf{f}$ is a program-function, the program's *principal identifier*.

Given a program-body P, we write $P \vdash E$ if E is equationally derivable from P, using reflexivity, substitution of terms for variables, and substitution of equals for equals.[15] Each (r-ary) program-function $\mathbf{g}$ in P induces on $\mathbb{A}$ the relation $\mathbf{g}^P =_{\mathrm{df}} \{(x_1 \ldots x_r, y) \mid P \vdash \mathbf{g}(x_1 \ldots x_r)\mathbf{=}y\}$. A (partial) function f over the algebra $\mathbb{A}$ is *computed* by $(P,\mathbf{f})$ if $f = \mathbf{f}^P$. For example,

[14] Herbrand–Gödel programs, a variant of equational programs for natural numbers, are defined e.g. in [Kleene, 1952].

[15] That is, (1) $P \vdash E$ for every $E \in P$; (2) $P \vdash x\mathbf{=}x$; (3) if $P \vdash E$ then $P \vdash [\mathbf{t}/u]E$ for every V_P-term $\mathbf{t}$ and variable u (where $[\mathbf{t}/u]$ denotes is the operation of substituting $\mathbf{t}$ for all (free) occurrences of u); and (4) if $P \vdash [\mathbf{t}/x](\mathbf{s}\mathbf{=}\mathbf{q})$ and $P \vdash \mathbf{t}\mathbf{=}\mathbf{t}'$, then $P \vdash [\mathbf{t}'/x](\mathbf{s}\mathbf{=}\mathbf{q})$. Note that taking x for $\mathbf{q}$ in (4) yields transitivity, and taking $x\mathbf{=}\mathbf{t}$ for $\mathbf{s}\mathbf{=}\mathbf{q}$ yields symmetry.

the program $(\{\ x + \mathbf{0}{=}x,\ \ x + \mathbf{s}(y){=}\mathbf{s}(x + y)\},\ \ +)$ over $\mathbb{N}$ computes the addition function, and $\{\ \epsilon \oplus w{=}w,\ \ (\mathbf{c}v) \oplus w{=}\mathbf{c}(v \oplus w)\ \},\ \ \oplus)$ over $\mathbb{W}$ computes the concatenation function.

We say that a program-body P is *coherent* if $P \vdash \mathbf{a} = \mathbf{a}'$ for no distinct $\mathbf{a}, \mathbf{a}' \in \mathbb{A}$.[16] A coherent P over $\mathbb{A}$, even if it computes a partial function, has a model containing $\mathbb{A}$ (i.e. where all program-functions are interpreted as *total* functions).

The convergence of equational programs can be expressed, without use of coding, by a second order validity statement, as follows. Let $\mathbb{A}$ be a free algebra, and let A be a second order definition of membership in $\mathbb{A}$, as in Section 2.3. Given a program-body P over $\mathbb{A}$, let $\tilde{\forall}P$ stand for the conjunction of the universal closures of all statements in the program P.

Theorem 2.5.1. *A coherent equational program $(P, \mathbf{f})$ over an algebra $\mathbb{A}$ computes a (unary) total function f iff*[17]

$$\models \tilde{\forall}P \wedge A[x] \rightarrow A[\mathbf{f}(x)]$$

Theorem 2.5.1 should be contrasted with the impossibility of capturing program convergence in first order logic:

Theorem 2.5.2. *Let $\mathbb{A}$ be an infinite free algebra. There is no effective procedure that assigns to every program $(P, \mathbf{f})$ over $\mathbb{A}$ a first order formula φ_P such that f is total iff $\models \varphi_P$.*

Proof. Suppose a procedure as above existed. The set of programs $(P, \mathbf{f})$ for which $\models \varphi_P$ is effectively enumerable, by the Completeness Theorem for first order logic. It is easy to see[18] that there is a recursive set C of programs such that every partial recursive function over $\mathbb{A}$ is computed by some program in C. Thus $\{\ f : \mathbb{N} \rightarrow \mathbb{N}\ \mid\ f$ is computed by some $P \in C$, where $\models \varphi_P\ \}$ is effectively enumerable and consists of all total computable functions, contradicting an elementary result of Computation Theory. ∎

2.6 Truth definitions: the analytical hierarchy

The arithmetical and the analytical hierarchies are well known classifications of sets of natural numbers. Here we give a self-contained outline of some relations of interest between these hierarchies, second order formulas, and truth-definitions.[19]

---

[16] For example, $\{\mathbf{f}(\mathbf{0}) = \mathbf{0}, \mathbf{f}(\mathbf{0}) = \mathbf{s0}\}$ is not coherent, nor is $\{\mathbf{g}(x) = \mathbf{f}(\mathbf{0}), \mathbf{f}(\mathbf{0}) = \mathbf{0}, \mathbf{f}(\mathbf{0}) = \mathbf{s0}\}$.

[17] Here $arity(\vec{x}) = arity(\mathbf{f})$, and $A(x_1 \ldots x_k)$ abbreviates $A(x_1) \wedge \cdots \wedge A(x_k)$.

[18] See e.g. [Kleene, 1952] for numeric programs, [Leivant, 1994] for programs over $\mathbb{W}$.

[19] We follow tradition in referring to $\mathbb{N}$, even though from a modern viewpoint richer free algebras, such as $\mathbb{W}$, are more appropriate.

A set $A \subseteq \mathbb{N}$ is **arithmetical** if it is first order definable over the standard structure of the natural numbers with all total computable functions.[20] The set A is **analytical** if it is second order definable over that structure. A fundamental result of Kleene is that there are analytical sets that are not arithmetical. It will be useful for us to discuss this fact from the viewpoint of *truth definitions*, since it brings out the interplay between increased expressive power and the formalization of meta-reasoning.

Let V be a countable vocabulary. We can code V-terms and V-formulas unambiguously by strings of ASCII characters,[21] for example, by writing `x_121` for the 121'st variable, `f^2_34` for the 34'th binary function-constant, and `(forall)` for the universal quantifier. Substituting octal ASCII codes for the ASCII characters, every syntactic expression α is unambiguously represented by a string $str(\alpha) = d_{\alpha,r(\alpha)} \ldots d_{\alpha,1} d_{\alpha,0}$ of digits $\in \{0,\ldots,7\}$, which in turn is unambiguously represented by the number $\#(\alpha) = 8^{r(\alpha)+1} + \sum_{i=0}^{r(\alpha)} 8^i \cdot d_i^\alpha$, whose octal numeral is 1 followed by $str(\alpha)$.

Clearly, there are primitive-recursive functions *neg*, *disj*, *univ*, *eq*, and *sbst* such that, for all formulas φ and ψ, variables v, terms t and s, and numbers n,[22]

$$\begin{array}{rclcrcl} neg(\#\varphi) & = & \#(\neg\varphi) & \qquad & disj(\#\varphi, \#\psi) & = & \#(\varphi \vee \psi) \\ eq(\#t, \#s) & = & \#(t{=}s) & & univ(\#\varphi, \#v) & = & \#(\forall v\, \varphi) \\ sbst(\#\varphi, n) & = & \#([\bar{n}/z]\varphi) & & \multicolumn{3}{l}{\text{where } z \text{ is a fixed variable-identifier}} \end{array}$$

Let $\mathcal{N}$ be an expansion of $\mathcal{N}_s$ with *finitely* many functions, including the primitive recursive functions above. Let L be a set of formulas in the vocabulary of $\mathcal{N}$. A **truth-definition** for L (over $\mathcal{N}$) is a formula $\tau \equiv \tau[v]$, with a single free variable v, such that $\mathcal{N} \models \varphi \leftrightarrow \tau[\overline{\#\varphi}]$ for all first order formulas φ in the vocabulary of $\mathcal{N}$.[23]

Theorem 2.6.1. [Tarski] *Let $\mathcal{N}$ be as above. Suppose L is a language such that the functions neg and sbst (or their graphs) are definable by L-formulas over $\mathcal{N}$. Then there is no truth-definition in L for L over $\mathcal{N}$.*

Proof. Suppose τ were a truth-definition in L for L over $\mathcal{N}$. Let $d(x)$ abbreviate $neg(sbst(x,x))$, and let $n =_{\mathrm{df}} \#(\tau[d(z)]) = \#(\tau[neg(sbst(z,z))])$. Then, in $\mathcal{N}$,

---

[20] By Gödel's proof of the First Incompleteness Theorem, it suffices to consider formulas using only 0,1,+ and ×.

[21] ASCII is the acronym for the common coding of characters and special symbols by 8 bits each.

[22] We write $[t/v]\varphi$ for the result of simultaneously substituting the term t for all free occurrences of v in φ, renaming bound variables as needed.

[23] We write $\varphi[t]$ for $[t/v]\varphi$ when v is unambiguous.

$$\begin{array}{lll} \tau[d(\bar{n})] \text{ i.e. } & \tau[neg\,(sbst\,(\bar{n},\bar{n}))] & \\ \leftrightarrow & \tau[neg\,(sbst\,(\overline{\#(\tau[d(z)])},\bar{n})] & \text{by definition of } n \\ \leftrightarrow & \tau[neg\,(\overline{\#(\tau[d(\bar{n})])}] & \text{by definition of } sbst \\ \leftrightarrow & \tau[\#(\overline{\neg\tau[d(\bar{n})]})\,] & \text{by definition of } neg \\ \leftrightarrow & \neg\tau[d(\bar{n})] & \text{since } \tau \text{ is a truth-definition} \end{array}$$

a contradiction. ∎

Theorem 2.6.2. *Let $\mathcal{N}$ be as above. There is a second order truth-definition, but no first order truth-definition, for first order sentences over $\mathcal{N}$. That is, the codes of first order sentences true in $\mathcal{N}$ form a set which is analytical but not arithmetical.*

Proof. The non-existence of a first order truth definition is a special case of Theorem 2.6.1.

To prove the existence of a second order truth definition let us start by observing that the truth of formulas is defined by recurrence on syntax, using (finite) valuations of free variables. A finite valuation in $\mathcal{N}$ can be represented by a number, using one of the usual sequence-coding methods. For any one of these methods there are primitive-recursive functions *proj* and *inst*, such that $proj\,(x,k)$ is the k'th entry of sequence coded by[24] x, and $inst\,(x,\#v,n)$ is the valuation that differs from x only in assigning n to the variable v.

A preliminary semantic evaluation assigns values to *terms*: It is easy to define a primitive-recursion function *val* such that $val(\#t,x)$ is the value of the term t under the assignment coded by x, i.e. such that $val(\#v,x) = proj\,(x,\#v)$ for variables v, and $val(\#(\mathbf{f}(t_1,\ldots t_r)),x) = f(val(\#t_1,x),\ldots, val(\#t_r,x))$ for every function $f = \mathbf{f}^{\mathcal{N}}$ of $\mathcal{N}$.[25]

We can now state that a set T contains a pair $(\#\varphi,h)$ iff φ is true under the valuation coded by h. If we use $\neg$, $\vee$, and $\forall$ as the only logical constants, it suffices to put:

$$\begin{array}{rcl} \tau_0[T] & \equiv & \forall u,v,h\,(T(eq\,(u,v),h) \;\leftrightarrow\; val\,(u,h) = val(v,h)) \\ & & \wedge\, \forall u,h\,(T(neg\,(u),h) \;\leftrightarrow\; \neg T(u,h)) \\ & & \wedge\, \forall u,v,h\,(T(disj\,(u,v),h) \;\leftrightarrow\; T(u,h) \vee T(v,h)) \\ & & \wedge\, \forall u,v,h\,(T(univ\,(u,v),h) \;\leftrightarrow\; \forall n\, T(u,\, inst(h,v,n))). \end{array}$$

A formula φ is true in $\mathcal{N}$ iff $\forall T\ (\tau_0[T] \rightarrow \forall h\, T(\overline{\#\varphi},h))$ is true in $\mathcal{N}$.

The definition of τ_0 uses some 8 primitive-recursive functions *neg* , *sbst* , ... Let $g_1 \ldots g_m$ be a list of all these functions and of the functions used

[24] with value 0 if k is not in the domain of x

[25] Recall that $\mathcal{N}$ has finitely many functions. The function *val* is, of course, not a function of $\mathcal{N}$. The treatment can be adapted to all primitive-recursive functions, as in [Troelstra, 1973], but the function *val* is then not primitive-recursive.

in their primitive recursive definitions. Let D be the conjunction of the (universal closure of the) primitive recursive definitions of $g_1 \dots g_m$. Then, for every first order sentence φ,

$$\mathcal{N} \quad \models \quad \varphi \leftrightarrow \tau(\overline{\#\varphi})$$

where

$$\tau(z) \quad \equiv_{\mathrm{df}} \quad \forall g_1 \dots g_m \, (D \rightarrow \forall T(\tau_0[T] \rightarrow T(z,0)) \,) \,).$$

∎

2.7 Inductive definitions

Inductive definitions have played a central role in the foundations of mathematics for over a century. They appear in a broad spectrum of topics, such as algebra (the subalgebra generated by a given set of elements), proof theory (the theorems of a theory, ordinal notations), descriptive set theory (Borel sets), and computation theory (the collection of functions generated by certain schemas, the 'structural-style' operational semantics of programs). In particular, they were used in the 1970's as the backbone of major generalizations of Recursive Functions Theory [Barwise, 1975; Moschovakis, 1974]. In recent years the ties between inductive definitions (in particular over finite structures) and database theory, descriptive computational complexity, and logics of programs, have been developed. We return to inductive definitions in sections 6.2 and 8.4.[26]

Let F be an operator over the collection $\mathcal{P}^r(U)$ of r-ary relations over some set U. F has an *inductive closure* F^∞, obtained as the union of the increasing ordinal-chain $F^\alpha =_{\mathrm{df}} \bigcup_{\xi<\alpha} F(F^\xi)$. Since $F^\alpha \subseteq F^\beta$ for $\alpha < \beta$, we must have $F^\infty = F^\alpha$ for a sufficiently large α. The first such α is the *closure ordinal* of the definition, and is $\leq \kappa^r$, where κ is the cardinality[27] of U. We then have $F(F^\infty) = F(F^\alpha) = F^{\alpha+1} = F^\infty$, i.e. F^∞ is a fixpoint of the operator F.

Suppose now that U is the universe of a structure $\mathcal{S}$, and that F is defined by a second order formula φ over the vocabulary of $\mathcal{S}$ expanded with an r-ary predicate letter R; that is, for some tuple $\vec{x}$ of distinct (free) variables,[28]

$$F(Q) = \{\vec{a} \mid [Q/R]\mathcal{S}, \, [\vec{a}/\vec{x}] \models \varphi\}.$$

We denote the operator F above by $\lambda R \lambda \vec{x} \varphi$. For example, the reflexive and transitive closure of a binary relation ρ is F^∞, where $F =_{\mathrm{df}} \lambda R \lambda x, y \, (x = y \vee \exists z \, R(x,z) \wedge \rho(z,y))$. For another example, the truth of first order

[26] A short informative survey is [Aczel, 1977]. A comprehensive monograph is [Moschovakis, 1974].

[27] the exponent r is needed because κ may be finite.

[28] $[Q/R]\mathcal{S}$ is the structure $\mathcal{S}$ expanded with Q as the interpretation of the relation-constant R.

arithmetic formulas using $\vee$, $\forall$, equations and inequalities[29] is $F^\infty[v,0]$, where

$$\begin{aligned} F =_{\text{df}} \lambda R\lambda z,x \qquad & \exists u,v\ \ z = eq\,(u,v) \wedge val(u,x) = val(v,x) \\ & \vee\ \exists u,v\ \ z = neg\,(eq\,(u,v)) \wedge val(u,x) \neq val(v,x) \\ & \vee\ \exists u\ \ z = disj\,(u,v) \wedge (R(u,x) \vee R(v,x)) \\ & \vee\ \exists u\ \ z = univ\,(u,v) \wedge \forall n\, R(u,\, inst(x,v,n)). \end{aligned}$$

The fixpoint F^∞ of a second order definable operator $F = \lambda R\lambda\vec{x}\varphi$ is also second order definable: Let $WF(Q)$ be a second order formula stating that Q is a well ordering of the universe in hand (such a formula can be defined easily, using ideas similar to Section 2.2). We have[30]

$$\vec{y} \in F^\infty \ \ \equiv_{\text{df}} \ \ \exists Q,S\ \ WF(Q) \wedge\ \forall\alpha\forall\vec{x}\ (\ S(\alpha,\vec{y}) \leftrightarrow \varphi_\alpha\ )\ \wedge\ \exists\alpha\ S(\alpha,\vec{y})$$

where φ_α is the result of replacing every subformula $R(\vec{t})$ of φ by $\exists\xi\ (\ Q(\xi,\alpha) \wedge S(\xi,\vec{t}))$.

The classical theory of inductive definability has traditionally focused on monotone operators. An operator F is *monotone* if $\rho \subseteq \rho'$ implies $F(\rho) \subseteq F(\rho')$. The inductive closure of a monotone operator F is its *least* fixpoint: if $F(\rho) = \rho$ then $\rho \supseteq F^\alpha$ for all ordinals α (by ordinal-induction), so $\rho \supseteq F^\infty$. That is, F^∞ is the intersection of all sets closed under F. This permits a much simpler second order definition of the inductive closure of a definable[31] monotone operator $F = \lambda R\lambda\vec{x}\ \varphi$:

$$\vec{x} \in F^\infty \quad \equiv_{\text{df}} \quad \forall R\ (\ (\varphi \equiv R)\ \rightarrow\ R(\vec{x})\ ).$$

It is easy to see that if all occurrences of R in φ are positive[32] then $\lambda R\lambda\vec{x}\,\varphi$ is monotone, and one writes $\mu R.\lambda\vec{x}\,\varphi$ (or, when in no danger of ambiguity, simply $\mu R.\varphi$), for the minimal fixpoint F^∞ of $F = \lambda R\lambda\vec{x}\,\varphi$.

For operators defined by *first order* formulas the converse also holds: if $\lambda R\lambda\vec{x}\,\varphi$ is monotone over *all* structures, where φ is first order, then there is a formula logically equivalent to φ in which all occurrences of R are positive (by Lyndon's Theorem, see e.g. [Chang and Keisler, 1973; Hodges, 1993]).[33]

---

[29] These restrictions are motivated by expository terseness, and could be removed.

[30] we use Greek letters as ordinary first order variables

[31] The expression $\varphi \equiv R$ abbreviates $\forall\vec{x}(\varphi \leftrightarrow R(\vec{x}))$, where $arity\,(\vec{x}) = arity\,(R)$.

[32] If an occurrence of ψ in φ is in the scope of n negations and in the negative scope of i implications, then ψ is *positive* in φ if $n+i$ is even.

[33] However, a first order operator may be monotone over all *finite* structures while failing to be positive [Ajtai and Gurevich, 1987]. The restriction to positive operators is therefore less natural in the context of Computer Science [Livchak, 1983; Gurevich, 1984].

3 Canonical semantics of higher order logic

3.1 Tarskian semantics of second order logic

Let us consider the semantics of second order formulas more formally. Let V be a vocabulary (as for first order logic), $\mathcal{S}$ a V-structure. Like for first order formulas, the truth of second order formulas in $\mathcal{S}$ is defined modulo a valuation η of variables as objects. Here, however, we have second order variables, to which one assigns the appropriate kind of objects. That is, a *valuation* η assigns structure elements to free object-variables, k-ary relations to free k-ary relation-variables, and k-ary functions to free k-ary function-variables ($k \geq 0$).

Terms' values are defined by recurrence:

$val_{\mathcal{S},\eta}(\mathbf{c})$	$=$	$\mathbf{c}^{\mathcal{S}}$	for object-constants $\mathbf{c}$
$val_{\mathcal{S},\eta}(v)$	$=$	$\eta(v)$	for object-variables v
$val_{\mathcal{S},\eta}(\mathbf{f}(t))$	$=$	$\mathbf{f}^{\mathcal{S}}(val_{\mathcal{S},\eta}(t))$	for $\mathbf{f}$ a unary function identifier of V and similarly for higher arities
$val_{\mathcal{S},\eta}(g(t))$	$=$	$\eta(g)(val_{\mathcal{S},\eta}(t))$	for g a unary function-variable, and similarly for higher arities.

The truth of atomic formulas is then defined by

$\mathcal{S},\eta \models t{=}t'$	iff	$val_{\mathcal{S},\eta}(t) = val_{\mathcal{S},\eta}(t')$
$\mathcal{S},\eta \models \mathbf{Q}(t)$	iff	$val_{\mathcal{S},\eta}(t) \in \mathbf{Q}^{\mathcal{S}}$ for $\mathbf{Q}$ a unary relation identifier of V, and similarly for higher arities
$\mathcal{S},\eta \models R(t)$	iff	$val_{\mathcal{S},\eta}(t) \in \eta(R)$ for R a unary relation-variable, and similarly for higher arities

Finally, the truth of compound formulas is defined by recurrence on formulas, as for first order formulas.

3.2 Function and relation formulations

Under the usual (classical, extensional) reading of functions and relations, these two concepts are of course reducible to each other: k-ary functions can be viewed as a special kind of $k+1$-ary relations, and k-ary relations can be interpreted by their k-ary characteristic functions. For instance, a formula $\exists f\,(\cdots X(f(t))\cdots)$ is interpreted by $\exists F\,(\forall x \exists! y F(x,y) \wedge (\cdots(\exists y(F(t,y) \wedge X(y))\cdots))$ (F fresh). Conversely, a formula $\forall X\,(\cdots X(t)\cdots)$ can be interpreted by $\forall f\,(\cdots(f(t) = \mathbf{c})\cdots)$, where $\mathbf{c}$ is a fixed constant identifier of V (and f is fresh).[34] It therefore suffices to

[34] We assume that the structure in hand has at least 2 elements.

formulate second order logic using only relation-variables, or only function variables.

On closer examination, however, the relational formulation of second order logic permits finer distinctions. An interesting case is the contrast between formulas of the form $\forall \vec{f}\,\exists \vec{x}\,\psi$ (ψ quantifier-free) and of the form $\forall \vec{R}\,\exists \vec{x}\,\psi$, over the standard structure of the natural numbers (with additional and multiplication). Whereas every Π^1_1 set in the analytical hierarchy is definable by some formula of the first form, the sets defined by formulas of the second form are all recursively enumerable [Kreisel, 1968]! The latter formulas were dubbed *strict*-Π^1_1 in [Barwise, 1969; Barwise, 1975], and *computational* in [Leivant, 1987]. The term 'computational' is motivated by the fact that each such formula defines, uniformly for all V-structures, the operational semantics of a certain finite state machine. They are, in many respects, more appropriate than Σ_1 formulas as a generalization, to arbitrary structures, of semi-recursiveness: the descriptive power of Σ^0_1 formulas over the natural numbers is derived from the possibility of numeric coding of computations; computational formulas describe computational processes without depending on such coding.[35]

3.3 Normal forms

The well-known procedure, for transforming each first order formula φ into a logically equivalent *prenex* formula (i.e. with no quantifier in the scope of a propositional connective), also transforms each second order formula into an equivalent prenex form. A second order prenex formula ψ can be further transformed (without increasing the number of quantifiers of any kind) into an equivalent formula ψ^* in which no second order quantifier falls in the scope of a first order quantifier. The transformation consists in recursively replacing subformulas of the form

$$\forall \vec{x}\forall f\ \varphi \quad \text{by} \quad \forall f\forall \vec{x}\ \varphi, \tag{3.1}$$

$$\exists \vec{x}\exists f\ \varphi \quad \text{by} \quad \exists f\exists \vec{x}\ \varphi, \tag{3.2}$$

$$\forall \vec{x}\exists f\ \varphi[\vec{x}, f] \quad \text{by} \quad \exists g\forall \vec{x}\ \varphi[\vec{x}, g_{\vec{x}}] \text{ where } g_{\vec{x}}(\vec{z}) =_{\mathrm{df}} g(\vec{x}, \vec{z}), \tag{3.3}$$

and

$$\exists \vec{x}\forall f\ \varphi[\vec{x}, f] \quad \text{by} \quad \forall g\exists \vec{x}\ \varphi[\vec{x}, g_{\vec{x}}], \tag{3.4}$$

and similarly for quantifiers over relation-variables. The two formulas in (3.3) are equivalent by (a weak form of) the Axiom of Choice, and (3.4) follows from (3.3) by duality.

Furthermore, every first order formula can be converted into an equivalent formula of the form $(\forall \vec{f})\,(\exists \vec{x})\,\psi$ or $(\exists \vec{f})\,(\forall \vec{x})\,\psi$, where ψ is quantifier-free, by replacing subformulas of the form

$$\forall \vec{x}\exists y\ \varphi[\vec{x}, y] \quad \text{by} \quad \exists f\forall \vec{x}\ \varphi[\vec{x}, f(\vec{x})],$$

[35] The significance of computational formulas is discussed further in [Leivant, 1991b].

$$\text{and} \qquad \exists \vec{x} \forall y \; \varphi[\vec{x}, y] \qquad \text{by} \qquad \forall f \exists \vec{x} \; \varphi[\vec{x}, f(\vec{x})].$$

and similarly for relational quantifiers.[36]

Combining these transformations, every second order formula φ can be converted into an equivalent formula φ^* of one of the forms $\forall \vec{\alpha}_1 \exists \vec{\alpha}_2 \forall \cdots Q \vec{\alpha}_m \, \overline{Q} \vec{x} \; \psi$ or $\exists \vec{\alpha}_1 \forall \vec{\alpha}_2 \exists \cdots Q \vec{\alpha}_m \, \overline{Q} \vec{x} \; \psi$, where $\overline{Q}$ is the dual of Q, each $\vec{\alpha}_i$ is a vector of second order variables, $\vec{x}$ a vector of element-variables, ψ is quantifier-free, and φ^* has at most one more function quantifier than φ. Formulas of these forms are said to be *normal*. A formula φ is Π^1_m if it is of the first form, Σ^1_m if it is of the second form.[37]

A well-known result of Kleene[38] is that for every $m > 0$ there is a set S_m of natural numbers defined by a Π^1_m formula but not by any Σ^1_m formula, and therefore also not by any Π^1_{m-1} formula. The set $\mathbb{N} - S_m$ is then defined by a Σ^1_m formula but not by any Π^1_m formula. In fact, it is easy to see that we can take for S_m the set of codes of Π^1_m sentences that are true in $\mathcal{N}$.

3.4 Finite order logic

Finite order logic (also called *Type Theory* and *ω-order logic*) is an extension of second order logic in which quantification is used over higher order relations and functions. It is basically due to Church [Church, 1949].[39] We outline a *relational* variant of finite order logic (similar to the account in [Schütte, 1960b, Chapter IV]).

Types (or, more precisely, *relational types*) are syntactic expressions inductively generated by: ι is a type; and if $\tau_1 \ldots \tau_k$ are types ($k \geq 0$), then $\tau = (\tau_1 \ldots \tau_k)$ is a type. The type () is denoted by o, and if $\tau_1 = \cdots = \tau_k = \iota$ then $(\tau_1, \cdots, \tau_k)$ is denoted by ι^k. The language of finite order logic has, for each type τ, variables of type τ and quantifiers over them.

We intend ι to denote the type (set) of individuals, i.e. structure elements, and $(\tau_1 \ldots \tau_k)$ the set of k-ary relations between k objects of types $\tau_1 \ldots \tau_k$, respectively. That is, for a set A, the set A_τ of objects of type τ over A is defined by

$$A_\iota \quad =_{\text{df}} \quad A,$$

[36] If the vocabulary V has an element-constant, then first order quantifiers can be replaced by function-quantifiers: $\forall x \psi[x] \equiv \forall f \psi[f(\mathbf{c})]$. Each first order formula has then equivalent formulas of both forms, $\forall \vec{f} \; \exists \vec{x} \; \chi$ and $\exists \vec{f} \; \forall \vec{x} \; \chi'$, where χ and χ' are quantifier-free.

[37] If a structure has a definable pairing function for structure-elements, as is the case for $\mathcal{N}_{+,\times}$ and its expansions, then each vector $\vec{\alpha}$ and $\vec{x}$ can be replaced by a single variable (with appropriate changes in the quantifier-free matrix).

[38] See e.g. [Kleene, 1952].

[39] A detailed textbook exposition is in [Andrews, 1986].

$$A_{(\tau_1 \ldots \tau_k)} \quad =_{\mathrm{df}} \quad \mathcal{P}(\prod_{i=1}^{k} A_i) = \mathcal{P}(A_1 \times \cdots \times A_k).$$

For type o we have $\prod_{i=1}^{0} A_i = \{()\}$ (a singleton), so A_o is the two element set $\{\emptyset, \{()\}\}$, whose elements can be identified with the boolean truth values *false* and *true*.

Finite order logic has τ-terms for each type τ, as follows.

1. a variable of type τ is a term of type τ;
2. if $\mathbf{f}$ is a k-ary function constant (of a given vocabulary), and $t_1 \ldots t_k$ are terms of type ι, then $\mathbf{f}(t_1 \ldots t_k)$ is a term of type ι;
3. if $\mathbf{R}$ is a k-ary relation-constant, then $\mathbf{R}$ is a term of type ι^k;
4. if t is a term of type $(\tau_1 \ldots \tau_k)$, and $t_1, \ldots, t_k$ are terms of types $\tau_1 \ldots \tau_k$, respectively, then $t(t_1, \ldots, t_k)$ is a term of type o.

Note that we do not use here defined, 'abstraction', terms. For example, if φ is a formula with a single free variable x, of type ι, then the set $\{x \mid \varphi\}$ is not accepted as a term of type (ι). To state that the property denoted by some variable Q of type $((\iota))$ hold of the set $\{x \mid \varphi\}$, one may write $\exists S^{(\iota)} \, (\forall x^{\iota} \, S(x) \leftrightarrow \varphi) \wedge Q(S)$, or $\forall S^{(\iota)} \, (\forall x^{\iota} \, S(x) \leftrightarrow \varphi) \rightarrow Q(S)$.

The terms of type o are the *atomic formulas,*[40] and compound formulas are generated as in first order and second order logic. The semantics of formulas of finite order logic is defined formally like for second order logic, except that an assignment over a structure with universe A maps variables of type τ to objects in A_τ.

Types naturally fall into *orders,* defined inductively by: $order(\iota) =_{\mathrm{df}} 1$, and $order((\tau_1, \ldots, \tau_k)) =_{\mathrm{df}} 1 + \max(order(\tau_1), \ldots, order(\tau_k))$. The fragment of finite order logic in which types of variables are of order $\leq k$ is dubbed **k'th order logic**. For $k = 1$ we recapture first order order logic, and for $k = 2$ we get the relational variant of second order logic.

The construction of types can be extended to allow types whose orders are transfinite ordinals. For example, let ω consist of all objects of finite type. Then (ω, ω) is the type of binary relations between objects of finite types. A formalism of transfinite types is developed and discussed in [Andrews, 1965] (see also [Montague, 1965a]).

3.5 Functional types

An alternative to relational types of the kind discussed above are functional types. **Functional types** are syntactic expressions inductively generated by: ι and o are types; and if σ and τ are types then so is $\sigma \rightarrow \tau$. The intent

[40] Equality between elements may be included among the relations of type ι^2.

is that $\sigma \to \tau$ is the type of functions from objects of type σ to objects of type τ.[41]

A function f of two arguments of type τ and returning a value of type σ can be represented by a function f_c of type $\tau \to (\tau \to \sigma)$, defined by $f_c(x)(y) = f(x, y)$. The mapping from f to f_c is called *currying*, in honour of Haskell Curry who discovered it. More generally, a function f of k arguments, of types $\tau_1 \ldots \tau_k$, and value of type σ, is represented by a function f_c of type $\tau_1 \to (\tau_2 \to \cdots \to (\tau_k \to \sigma) \cdots)$. The latter type is often abbreviated by $\tau_1, \ldots, \tau_k \to \sigma$.

The well-known duality between functions and relations over basic objects extends to the entire type hierarchy. Every relation R of relational-type $(\tau_1 \ldots \tau_k)$ can be thought of as a function which returns boolean values, i.e. R is represented by the function χ_R of type $\tau_1, \ldots, \tau_k \to o$, defined by

$$\chi_R(a_1 \ldots a_k) = \textbf{if } R(a_1 \ldots a_k) \textbf{ then } \mathit{true} \textbf{ else } \mathit{false}.$$

Conversely, a function f of type $\tau_1, \ldots, \tau_k \to \sigma$ is represented by its graph, which is of type $(\tau_1, \ldots, \tau_k, \sigma)$.

In every higher type calculus the interaction between expressions denoting objects of different types is governed by the operations of application and abstraction. If **a** and **b** denote objects of types $\tau \to \sigma$ and τ, respectively, then the applicative expression $(\mathbf{a})(\mathbf{b})$ denotes an object of type σ, namely the result of applying the object denoted by **a** to that denote by **b**. If **a** is an expression denoting an object of type σ, possibly parametrized by a variable x ranging over objects of type τ, then the abstraction-expression $\lambda x.\, \mathbf{a}$ denotes an object of type $\tau \to \sigma$, namely the function that to every object c of type τ assigns the value of **a** when the variable x is evaluated as **c**.

3.6 Formulas as higher order functions

Propositional logic can be construed as a boolean algebra, whose objects, the truth values, are of type o. The algebra's functions are the propositional connectives; negation is a function of type $o \to o$, and the binary connectives are of type $o, o \to o$. Propositional formulas are simply algebraic expressions, which define functions over truth values; for instance, the formula $(p \wedge \neg q) \to r$ is the function that for boolean arguments b_p, b_q and b_r return the boolean value $(b_p \wedge \neg b_q) \to b_r$.

Church [1949] showed how this simple functional interpretation of propositional logic can be extended to first order logic, using higher order functions. Whereas propositional identifiers denote truth values, relational

[41] An alternative notation to $\sigma \to \tau$, common in works about type theory, is $\tau(\sigma)$ [Church, 1949].

identifier denote boolean-valued functions over the type ι of individual objects. For example, the formula $\varphi \equiv P(x) \wedge \neg Q(y)$ is represented by a function F of type $\iota, \iota, (o \to \iota), (o \to \iota) \to o$, namely the function that for arguments a_x, a_y, F_P, and F_Q returns the value $F_P(a_x) \wedge \neg F_P(a_y)$. The order of arguments is arbitrary here. Alternatively, we may think of φ as a template from which functions can be defined by explicit abstraction. That is, $\lambda x\, \varphi$ denotes a function of type $\iota \to o$, parametrized by the identifiers y, P and Q. For each appropriate choice of values for these identifiers, $\lambda x\, \varphi$ denotes a function from objects to truth values.

Under this reading, quantifiers are higher order functions. Let $\forall$ be a function of type $(\iota \to o) \to o$, such that $\forall f =$ **if** [f is identically *true*] **then** *true* **else** *false*. Then $\forall(\lambda x \varphi)$ is *true* iff φ is *true* for every value of x. Thus, the usual first order notation $\forall x.\, \varphi$ can be viewed as an abbreviation for $\forall(\lambda x \varphi)$, where $\forall$ is the functional above. The existential quantifiers and many other quantifiers used in special logics (such as 'there are infinitely many') can similarly be interpreted as functions of type $(\iota \to o) \to o$.

Higher order quantifiers fall into the same mould. Consider universal quantification over unary relations, as in the formula $\forall P.\, \varphi$, where φ is as above. This can be viewed as an abbreviation for the functional expression $\forall^2(\lambda P.\, \varphi)$. The type of $\lambda P.\, \varphi$ is $(\iota \to o) \to o$, so the type of the operator $\forall^2$ used here is $((\iota \to o) \to o) \to o$. More generally, universal quantification over objects of type τ is an operator $\forall^\tau$ of type $(\tau \to o) \to o$. To define a generic universal quantifier that can be used for all types one needs to enrich the type structure beyond the one we have been discussing (cf. [Girard, 1972; Coquand and Huet, 1985]). Namely, in Girard's system F_ω, the generic universal quantifier is an operator **forall** of type $\Delta t.\, (t \to o) \to o$, so that **forall**$(\tau)$ is functionally equivalent[42] to $\forall^\tau$.

Church's interpretation of formulas as higher type functions is used extensively in Computer Science in formal calculi such as the Calculus of Constructions [Coquand and Huet, 1985], as well as in programming systems such as Edinburgh LEGO, L. Paulson's ISABELLE [Paulson, 1989; Paulson, 1990], Andrews's TPS [Andrews *et al.*, 1984], and Miller's $\boldsymbol{\lambda}$-Prolog [Nadathur and Miller, 1988; Nadathur and Miller, 1990; Miller, 1993].

3.7 Truth definitions revisited

It is rewarding to relate finite order logic to the issue of truth definitions mentioned in Section 2.6 above. The proof of Theorem 2.6.2 can be easily adapted to higher order formulas:

[42] I.e., **forall**(τ) reduces to $\forall^\tau$ by a type β-reduction. Note, however, that the generic **forall** is not a global quantifier that ranges simultaneously over objects of all types.

Theorem 3.7.1.

1. *Let $\mathcal{N}$ be an expansion of $\mathcal{N}_s$ in which the functions neg and sbst, as well as an injective pairing function p with its inverse projections, are definable.*
2. *There is no second order truth-definition for second order sentences over $\mathcal{N}$, but there is a third order such truth-definition.*

Proof. The absence of a second order truth definition is analogous to Theorem 2.6.2(1).

To define a third order truth definition let us first assume that all second order variables are set-variables (i.e. unary relation-variables). This is no loss of generality, since the presence of a pairing function p and its inverse projections permits the replacement of relations of higher arities by sets.

We would like to code by a set of natural numbers an assignment (into $\mathbb{N}$) of sets to a finite number of variables. If η is such an assignment, with domain $\{v_1 \ldots v_k\}$, then η can be coded by the set $H = \bigcup_{i=1}^{k}(\{\#(v_i)\} \times \eta(v_i))$, where $(\{n\} \times X) =_{\mathrm{df}} \{p(n,x) \mid x \in X\}$. Modify the primitive-recursive functions used in the definition of τ_0 in the proof of Theorem 2.6.2, to apply to formulas with second order quantification, and to valuations for second order logic. Clearly, there are primitive-recursive functions *type* and *apply*, such that for every variable v, $type(\#v) =$ **if** v is a number-variable **then** 0 **else** 1; and for any term t and set-variable V, $apply(\#V, \#t) = \#(V(t))$. Emulating the definition of τ_0 in the proof of Theorem 2.6.2(1), we can write down a formula τ_0^2, as follows, which states that a set T contains a triplet $(\#\varphi, h, H)$ iff φ is true under the valuation whose set-valued part is coded by H, and whose number-valued part is coded by h.

$$
\begin{aligned}
\tau_0^2[T] \quad \equiv_{\mathrm{df}} \quad & \forall u, v, h, H \, (\, T(eq(u,v), h, H) \; \leftrightarrow \; val(u,h) = val(v,h) \,) \\
& \wedge \, \forall u, v, h, H \, (\, type(u) = 1 \rightarrow \\
& \qquad (\, T(apply(u,v), h, H) \; \leftrightarrow \; H(p(u, val(v,h))))) \\
& \wedge \, \forall u, h, H \, (T(neg(u), h, H) \; \leftrightarrow \; \neg T(u, h, H)) \\
& \wedge \, \forall u, v, h \, (T(disj(u,v), h, H) \; \leftrightarrow \; T(u, h, H) \vee T(v, h, H)) \\
& \wedge \, \forall u, v, h, H \, (type(v) = 0 \rightarrow \\
& \qquad (\, T(univ(u,v), h, H) \; \leftrightarrow \; \forall n \, T(u, inst(h, v, n), H))) \\
& \wedge \, \forall u, v, h, H \, (type(v) = 1 \rightarrow \\
& \qquad (\, T(univ(u,v), h, H) \; \leftrightarrow \\
& \qquad\quad \forall Y \, (\, (Y = H \bmod (v)) \rightarrow T(u, h, Y))))
\end{aligned}
$$

where $Y = H \bmod (v)$ abbreviates

$$\forall u, Z\,(u \neq v \rightarrow\ ((u \times Z) \subset H\ \leftrightarrow\ (u \times Z) \subset Y)).$$

A second order sentence φ is then true iff the third order sentence $\forall T(\tau_0[T] \rightarrow\ T(\overline{\#\varphi}, 0, \emptyset))$ is true. (Note that T is a third order variable.)

To keep the definition above within the relational variant of finite order logic, we need to reformulate τ_0^2, but this is fairly straightforward. The proof is concluded as for Theorem 2.6.2. ∎

More generally, we have:

Theorem 3.7.2. *Let $\mathcal{N}$ be as above, $k \geq 1$.*

1a. There is no k order truth-definition for k order sentences over $\mathcal{N}$.

1b. There is no finite order truth-definition for all finite order sentences over $\mathcal{N}$.

2. There is a $(k+1)$ order truth-definition for k order sentences over $\mathcal{N}$.

A similar method can be used to show that there is a $(k+1)$ order formula τ such that for every k order sentence φ, $\tau[\overline{\#\varphi}]$ is valid iff φ is valid.[43]

The construction of truth definitions by using ever higher types can be continued to transifinite types: [Andrews, 1965] shows how to give a truth definition over $\mathcal{N}$ for all finite order formulas, using quantification over objects of transifinite type.

4 Proof theory

4.1 Basic formalisms

By Gödel's Completeness Theorem, provability in first order logic captures precisely validity. This success contrasts with second (and higher) order logic:[44]

Theorem 4.1.1. *The set of valid second order formulas is not definable (under canonical numeric coding) by a second order formula in the language of arithmetic. It is therefore not recursively enumerable; in particular, there is no effective formalism whose theorems are precisely the valid second order formulas.*

Proof. Let ν be the conjunction $\varphi_N \wedge D_{+\times}$ (see Section 2.3). The models of ν have the natural numbers as domain and the standard interpretation for $+$ and $\times$. Therefore, a formula ψ in the language of Peano's Arithmetic[45]

[43] Note that for $k = 1$ such a formula falls out from the Completenss Theorem for first order logic, as follows. let Pr be an existential formula of arithmetic that formalizes provability: for every first order formula φ, $Pr(\overline{\#\varphi})$ iff φ is provable. Then φ is valid iff it is provable, iff $Pr(\overline{\#\varphi})$ is true in the standard model, iff $\varphi_N \wedge D_{+\times} \rightarrow Pr(\overline{\#\varphi})$ is valid.

[44] A stronger result is Theorem 5.6.3 below.

[45] i.e. over the vocabulary $\{\mathbf{0}, \mathbf{s}, +, \times\}$

is true (in the standard model) iff the formula $\nu \rightarrow \psi$ is valid. By Theorem 2.6.1 the set of true second order formulas of arithmetic is not second order definable. Thus the set of valid formulas of the form $\nu \rightarrow \psi$ is not second order definable by a second order arithmetic formula. Since the formulas of the form $\nu \rightarrow \psi$ are effectively recognizable, it follows that the set of *all* valid second order formulas is also not definable by an arithmetic second order formula. ∎

Though second order logic does not have a complete deductive calculus, it does have sound deductive calculi that are logically natural, powerful, of metamathematical interest, and suitable for the formalization of much of mathematics, of computer science, and of cognitive science. Moreover, these formalisms satisfy important syntactic properties[46] which permit a natural adaptation to these formalisms of methods of automated theorem proving [Andrews *et al.*, 1984].

The expressive power of second order logic stems from the standard interpretation of the second order quantifiers as ranging over *all* relations and functions over the structure in hand. It is not surprising that this range cannot be enforced by a deductive formalism, since each such formalism, being countable, can distinguish (implicitly or explicitly) only between a countable number of functions and relations (recall the Downward Skolem-Löwenheim Theorem).[47]

A natural formalism L_2 for the relational variant of second order logic uses the usual axioms for first order logic, with quantifier rules applying to relational variables as well as individual variables (we give a precise formulation momentarily), with the stipulation that the range of relation-variables includes *at least* all the relations definable by the formulas of the language. This stipulation is a form of the Comprehension Principle of Set Theory, and is formally rendered by the schema:

$$\exists R \, \forall x_1 \ldots x_k \; (\; R(x_1 \ldots x_k) \leftrightarrow \varphi \;)$$

where $k \geq 0$, R is a k-ary relation-variable, and φ is a second order formula in which R does not occur free.[48]

The formalism L_2 can be spelled out within any one of the familiar proof styles used for first order logic. A Hilbert-style calculus for L_2 is obtained by augmenting the inference rules and axiom schemas of first order logic with

[46] such as normalization for natural deductions, cut-elimination for sequential proofs, and certain forms of the subformula property; see Section 4.4)

[47] This plausibility argument is not a proof, of course. A related phenomenon is Trakhtenbrot's Theorem, that the collection of first order formulas true in all finite structures is not RE [Trakhtenbrot, 1950; Ebbinghaus *et al.*, 1984].

[48] Note that φ may have free variables other than $x_1 \ldots x_k$, and need not have all of $x_1 \ldots x_k$ free. Allowing R free in φ is a strong form of circular definition, and leads to contradictions: take, e.g., $k = 0$ and $\varphi \equiv \neg R$.

the Comprehension Schema above, and quantification rules for the relation-variables:[49] the axiom schema of universal instantiation: $(\forall R\,\varphi) \rightarrow [Q/R]\varphi$ ($arity(R) = arity(Q)$), and the inference rule of universal generalization:

$$\frac{\psi \rightarrow [Q/R]\varphi}{\psi \rightarrow \forall R\,\varphi} \qquad Q \text{ not free in } \psi$$

A natural-deduction calculus for L_2 is obtained by supplementing the inference rules of first order logic with quantification rules for relation-variables. Writing $\Gamma \Rightarrow \varphi$ for 'φ is derived from assumptions Γ', we have the following rules for second order $\forall$-introduction and $\forall$-elimination:

$$\forall\mathrm{I}\ \frac{\Gamma \Rightarrow [Q/R]\varphi}{\Gamma \Rightarrow \forall R\,\varphi} \qquad Q \text{ not free in } \Gamma$$

and

$$\forall\mathrm{E}\ \frac{\Gamma \Rightarrow \forall R\,\varphi}{\Gamma \Rightarrow [\lambda\vec{x}.\psi/R]\varphi}$$

where $arity(\vec{x}) = arity(R) = arity(Q)$, ψ is free for R in φ,[50] and $[\lambda\vec{x}.\psi/R]\varphi$ is the result of replacing every subformula $R(\vec{t})$ of φ by $[\vec{t}/\vec{x}]\psi$. Note that $\forall$E conveys the Comprehension Principle.

A sequential-style calculus for L_2 is defined similarly.[51]

Moving on from second order to finite order logic, it is obvious how to lift the Comprehension Principle to types of higher order. If $x_1 \ldots x_k$ are variables of types $\tau_1 \ldots \tau_k$ respectively, $\tau = (\tau_1 \ldots \tau_k)$, and R is a variable of type τ, then **comprehension at type** τ is the schema

$$\exists R\ \forall x_1 \ldots x_k\ (\ R(x_1 \ldots x_k) \leftrightarrow \varphi\ ) \qquad R \text{ not free in } \varphi.$$

We denote by L_n the formalism of n order logic with comprehension for all types of order $\leq n$, and by L_ω finite order logic with comprehension at all types.

4.2 Additional set existence principles

Comprehension is a purely logical, syntax driven, set existence principle. It encompasses all elementary set existence axioms of set theory to the extent that they pertain to second order logic: the Empty-Set, Union, and Separation axioms are instances of Comprehension, whereas a form

[49] Such a formalism is detailed, for example, in [Church, 1956, Section 50]. Note that the axiom schema and rules, including the ones of first order logic, now apply to all formulas in the language. Analogous axiom-schema and inference-rule should be added for $\exists$, unless $\exists$ is viewed as an abbreviation for $\neg\forall\neg$.

[50] for both element- and relation-variables, in the usual sense

[51] See for example [Takeuti, 1975; Miller *et al.*, 1991]. A resolution-style calculus for higher order Horn clauses can be found in [Miller, 1993].

of Pairing is present in the syntax of L_2, which has relation-variables for any finite arity. The Power-Set axiom is outside the scope of second order logic, since it collapses the entire type hierarchy into a first order theory. However, in finite order logic we do have a weak form of the Power-Set Principle: if X is a set of order k, then the power-set of X is a legitimate set of order $k+1$, since by Comprehension we have $\exists P\, \forall Y\, (\, P(Y) \leftrightarrow Y \subseteq X\,)$. The Axiom of Infinity is orthogonal to finite order logic, since the latter is a formalism for *all* structures, including finite ones.

A set-existence principle of interest is Choice, which in L_2 can be formulated as the single axiom:[52]

$$\forall D\, (\, \forall x \exists y\, D(x,y) \rightarrow \exists R^2\, \forall x \exists! y\, (\, R(x,y) \wedge D(x,y)\,)$$

Choice can be lifted to arbitrary types: Choice for types σ, τ reads:[53]

$$\forall D^{(\sigma,\tau)}\, (\, \forall x^{\sigma} \exists y^{\tau}\; D(x,y) \rightarrow \exists R^{(\sigma,\tau)}\, \forall x^{\sigma} \exists! y^{\tau}\; R(x,y) \wedge D(x,y)\,).$$

For types (ι, σ), with $\sigma = (\sigma_1 \ldots \sigma_r)$, we can reformulate this axiom as a principle of Collection [Hilbert and Ackermann, 1928, Section IV.1]:

$$\forall D^{(\iota,\sigma)}\, (\, \forall x^{\iota} \exists y^{\sigma}\; D(x,y) \rightarrow \exists z^{(\iota,\sigma)}\, \forall x^{\sigma}\, D(x, z_x)\,),$$

where

$$D(x, z_x) \;\; \equiv_{\mathrm{df}} \;\; \exists y^{\sigma}\; (\, \forall u_1^{\sigma_1} \ldots u_r^{\sigma_r}\; (\, y(\vec{u}) \leftrightarrow z(x, \vec{u})\,) \;\wedge\; D(x,y)\,).$$

This axiom gives rise to a Collection *Schema* at lower orders. In particular, for $\sigma = (\iota)$ we get a schema of choice from objects to sets, in second order (not third order) logic:

$$\forall x \exists Y\, \varphi[x, Y] \rightarrow \exists Z\, \forall x \varphi[x, Z_x],$$

where $\varphi[x, Z_x]$ is $\varphi[x, Y]$, with every subformula $Y(t)$ replaced by $Z(x,t)$. Of some interest are variants of the Choice principles above, such as the schema of Dependent Choice and Generalized Dependent Choice (see e.g. [Kreisel, 1968; Feferman, 1977]).

---

[52] $\exists!$ abbreviates 'there is a unique': $\exists! y\, \psi \;\; \equiv_{\mathrm{df}} \;\; \exists y\, (\psi \wedge \forall z(\, [z/y]\psi \rightarrow z = y\,))$. There are alternative formulations, all equivalent by the use of weak forms of Comprehension. [Hilbert and Ackermann, 1928] gives the following schema, which states that for every binary relation D there is a function R that acts as a choice (partial) function on the domain of D: $\forall D \exists R\, (\, \forall x, y, z\, (R(x,y) \wedge R(x,z) \rightarrow y \mathbf{=} z) \wedge \forall x\, (\exists y\, D(x,y) \rightarrow \exists y\, (\, R(x,y) \wedge D(x,y)\,)\,)$. The following form states that every appropriate binary D contains a choice function: $\forall D (\, \forall x \exists y\, D(x,y) \rightarrow \exists R^2\, (\forall x \exists! y (\, R(x,y) \wedge D(x,y)) \wedge \forall x, y\, (R(x,y) \rightarrow D(x,y))\,)$.

[53] Here equality for type τ, which is used in spelling out $\exists y^{\tau}$, can be defined by recurrence on τ from equality at type ι.

Another set-existence principle is Fraenkel's schema of Replacement, which states that the image of a set under a class-function is a set. In the context of second and finite order logic, class-functions are admitted as relations (by Comprehension), and Replacement is therefore deducible.

Other set-existence principles in set theory, such as high-cardinal axioms, GCH, the existence of measurable cardinals, the diamond principle, or the axiom of determinacy, can be formulated in second order logic as statements about sufficiently large structures, but these principles are increasingly alien to the spirit of finite order logic.

4.3 Constructive finite order logics

At first blush, constructive (intuitionistic) finite order logic might look like a dubious hybrid: on the one hand one weakens even propositional rules to meet the demands of a restrictively constructive ontology, while on the other hand one brings in higher order quantification, whose ontology is problematic even classically. However, the combination of constructive logic and higher order quantification is conceptually natural, because higher order quantification does not depend on the legitimacy of the full power-set construction (compare Section 5.4). Constructive finite order logic is in fact a highly fecund formalism, with strong and useful ties to computing and to programming. Perhaps the most powerful ingredient of that connection is the close resemblance, discovered by Howard [Howard, 1980], between typed lambda calculi and natural deduction calculi (see Section 8.2).

It is well-known that constructive first order logic is a formalism at least as rich as classical first order logic, since the latter is interpretable in the former.[54] For instance, if φ is a first order formula, and φ' arises from φ by double-negating every atomic, disjunctive, and existential subformula of φ,[55] then $\varphi \leftrightarrow \varphi'$ is provable in classical logic, and φ is provable in classical logic iff φ' is provable in constructive logic.

Constructive (intuitionistic) variants, $I\!L_n$, of the formalisms L_n ($2 \leq n \leq \omega$) are obtained from the corresponding classical proof calculi as for first order logic. For instance, a sequential style calculus for $I\!L_2$ is the same as a calculus for L_2, but with all succedents restricted to have at most one formula. We then obtain an interpretation of L_2 in $I\!L_2$: for a formula φ let φ' be defined as above (with relational existential-quantifiers also double-negated). Then $\varphi \leftrightarrow \varphi'$ is provable in L_2, and φ is provable in L_2 iff φ' is provable in $I\!L_2$.

The richness of constructive formalisms compared to their classical counterparts has numerous manifestation. One striking example, for higher

---

[54] [Kolmogorov, 1925; Gödel, 1932]; see also [Friedman, 1978; Dragalin, 1979; Leivant, 1985]; textbook expositions can be found in [Kleene, 1952, Section 81] and [Troelstra, 1973].

[55] I.e, $\varphi' \equiv \neg\neg\varphi$ for atomic φ $(\psi \wedge \chi)' \equiv \psi' \wedge \chi'$, $(\psi \to \chi)' \equiv \psi' \to \chi'$, $(\psi \vee \chi)' \equiv \neg\neg(\psi' \vee \chi')$, $(\forall x\, \psi)' \equiv \forall x\, \psi'$, and $(\exists x\, \psi)' \equiv \neg\neg\exists x\, \psi'$.

order logic, was discovered by M. H. Löb [1976]. Let L_2^0 be classical second order propositional logic,[56] that is, the fragment of L_2 in which all relations are 0-ary. It is well-known that the truth of an L_2^0 formula φ (which holds iff φ is provable in L_2) is decidable by ana lgorithm that uses fairly manageable computational resources [57] Let now IL_2^0 be the 0-ary fragment of IL_2. Then provability in IL_2^0 is undecidable altogether: indeed, there is a reduction of provability in classical first order logic to provability in IL_2^0.[58]

An interesting property of first order constructive logic is the independence therein of the usual logical constants $(\neg, \wedge, \vee, \rightarrow, \forall, \exists)$. That is, no one of these constants can be defined in terms of the remaining ones, in contrast to classical first order logic. This is no longer the case with second order logic: Prawitz [1965] observed that all logical constants are definable in constructive second order logic in terms of $\rightarrow$ and $\forall$ we have:

$$\begin{array}{rcl} \mathit{false} \equiv - & \equiv & \forall X\, X \\ \neg\varphi & \equiv & \varphi \rightarrow - \\ \varphi \wedge \psi & \equiv & \forall X\, ((\varphi \rightarrow (\psi \rightarrow X)) \rightarrow X) \\ \varphi \vee \psi & \equiv & \forall X\, ((\varphi \rightarrow X) \rightarrow ((\psi \rightarrow X)) \rightarrow X)) \\ \exists x\, \varphi & \equiv & \forall X\, \forall x\, ((\varphi \rightarrow X) \rightarrow X) \\ \exists R\, \varphi & \equiv & \forall X\, \forall R\, ((\varphi \rightarrow X) \rightarrow X). \end{array}$$

Furthermore, the definability of $-$ implies that constructive second order logic is interpretable in *minimal* second order logic,[59] in which the constructive rule for negation, i.e. the schema $- \rightarrow \varphi$, is absent.

4.4 Normalization and the subformula property

A central topic of Proof Theory is *normalization* of natural-deductions, or *cut-elimination* for sequential proofs. Normalization is the transformation of a natural-deduction derivation into an equivalent derivation[60] which is *normal,* that is without any detour, where a *detour* is a formula which is derived by an *introduction* rule, and is also the principal premise of an *elimination* rule.[61] For example, in a deduction of the form

[56] i.e. the proof formalism for boolean quantified formulas.

[57] [Stockmeyer, 1974]; see e.g. [Hopcroft and Ullman, 1979]; the decidability of quantified boolean formulas is a canonical example of a poly-space complete problem.

[58] This complexity of IL_2^0 is related to the facts that there is no truth-table semantics for which intuitionistic propositional logic is sound and complete, but there is such a semantics with a countable set of truth values. Thus, boolean quantification in IL_2^0 may be seen as ranging over an infinite countable set.

[59] also called *positive* second order logic

[60] Derivations are equivalent if they prove the same formula from the same (open) assumptions.

[61] This definition of normal derivation suffices in the absence of $\vee$ and $\exists$.

$$\dfrac{\dfrac{\begin{array}{c}[\psi]\\ \Pi\\ \varphi\end{array}}{\psi\to\varphi}\qquad\begin{array}{c}\Sigma\\ \psi\end{array}}{\varphi}$$

the displayed formula $\psi\to\varphi$ is a detour. It can be eliminated by transforming the proof above into a more direct proof of φ:

$$\begin{array}{c}\Sigma\\ [\psi]\\ \Pi\\ \varphi\end{array}$$

Some of the indicated occurrences of the formula ψ may be new detour formulas, but these are shorter than the detour formula $\psi\to\varphi$ which has been eliminated.

Analogously, a detour may involve a second order generalization ($\forall$I) followed by an instantiation ($\forall$E):

$$\dfrac{\dfrac{\begin{array}{c}\Pi\\ [Q/R]\varphi\end{array}}{\forall R\,\varphi}}{[\lambda\vec{x}.\chi\,/\,R]\varphi}$$

This can be eliminated by a relational substitution in the derivation Π:

$$\begin{array}{c}[\lambda\vec{x}.\chi/Q]\;\Pi\\ [\lambda\vec{x}.\chi/R]\varphi\end{array}$$

However, this transformation might replace one detour by larger ones: if a formula ξ is a detour within Π, then the potentially larger formula $[\lambda\vec{x}.\chi/Q]\xi$ might be a detour within $[\lambda\vec{x}.\chi/Q]\Pi$.[62] Using a powerful combinatorial method due to Girard [1972], Prawitz [1972] proved that, in spite of this difficulty, every natural-deduction derivation of L_2 can be transformed effectively into a normal derivation, and Martin-Löf [1973a] showed that this hold for full finite order logic.[63]

[62] True, if Π is normal, then no such danger exists, but this potential for blow-ups in the complexity of detours ruins the use of syntactic size of detour formulas as a yardstick for progress on eliminating all detours: in eliminating the implicational detour above, ψ might be a detour of the form $\forall R\psi_0$, whose elimination would result in detours of syntactic complexity greater than that of the original detour $\psi\to\varphi$.

[63] See [Gallier, 1990] for a survey. Earlier proofs were given for the somewhat weaker

The fact that every derivation of first order logic can be normalized has far reaching applications, many of which are related to the *subformula property:* in a normal proof Π of a formula φ, every formula is a *subformula* of φ. The notion of subformula here is modulo term-substitution; for instance, the subformulas of $\forall x\varphi$ include all formulas $[t/x]\varphi$, where t is a term free for x in φ. Consequently, formulas in a normal proof have logical complexity bounded by the complexity of the derived formula, a useful fact in structural analyses of proofs for metamathematical applications.

Normal derivations of L_2 also have a subformula property, but the notion of a subformula is again modulo substitutions, which this time applies to second order quantifiers as well: the subformulas of $\forall R\,\varphi$ include all formulas of the form $[\lambda\vec{x}\psi/R]\varphi$, where ψ is free for R in φ. Thus, the logical complexity of formulas in a normal proof is no longer bounded by the complexity of the derived formula, and the subformula property for second order logic has limited metamathematical applications.[64]

5 Ontology

5.1 The gulf between first order and second order logic

First order logic has important properties lacking of second order logic. We enumerate in this section a few of them, before considering some foundational consequences of these differences.

1. The set of valid first order sentences is recursively enumerable, whereas the set of valid second order sentences is not definable in second order arithmetic (Theorem 4.1.1), nor even in finite order arithmetic (Theorem 5.6.3).
2. A logic L is *compact* if a set Γ of formulas has a model whenever each finite subset of Γ has a model. First order logic is compact, whereas second order logic is not: Let Γ consist of $\forall xN[x]$ and of all formulas $\mathbf{c} \neq \bar{n}$ for $n \geq 0$. Then every finite $\Gamma_0 \subset \Gamma$ is true in the structure $\mathcal{N}_s$, expanded with an interpretation of $\mathbf{c}$ as a sufficiently large natural number. But Γ has no model.
3. A logic L has the *Downward Skolem–Löwenheim Property* if every countable set of formulas that has an infinite model has a countable model. First order logic has this property, but second order logic

Normal Form Theorem, which states that every provable formula has a normal proof. Tait proved this property for second order logic [Tait, 1966], and Takahashi [1967] and Prawitz [1968] independently proved it for full finite order logic. These proofs use a model theoretic method of 'partial valuation' due to Schütte [1960a].

[64] This is less true for certain fragments of second order logic, in which the complexity of substituted formulas can be effectively controlled [Nadathur and Miller, 1990; Miller *et al.*, 1991].

does not: the sentence $\neg(\sigma_{<\aleph_1})$ (see Section 2.2) has a model, but no countable model.

4. A logic L has the *Upward Skolem–Löwenheim Property* if every countable set of formulas that has an infinite countable model has arbitrarily large models. First order logic has this property, but second order logic does not: the sentence $\sigma_{<\aleph_1}$ has an infinite countable model, but no uncountable model.
5. A generalized form of the downward Skolem–Löwenheim, which applies to any logic L, is as follows: there is a function F on the cardinals such that, if L has a model of size $> \kappa$, then it has a model of size $\kappa' \leq F(\kappa)$. The proof of the Downward Skolem–Löwenheim property for first order logic gives $F(\kappa) = \kappa + \aleph_0$. For second order logic even the function $F(\kappa) =_{\text{df}}$ the κ'th inaccessible cardinal (or even the κ'th measurable cardinal, assuming these cardinals exist) will not do [Barwise, 1972, Theorem 2.1].
6. First order logic satisfies Beth's *definability property,*[65] but second (and higher) order logic does not. A logic L has the definability property if each implicitly definable relation is explicitly definable, in the following sense. Suppose V is a vocabulary, and $V' = V \cup \{\mathbf{R}\}$, where $\mathbf{R}$ is a fresh relational identifier. Suppose Γ is a V'-theory, so that every V-structure can be expanded into a model of Γ in at most one way[66]; then there is a V-formula γ that is coextensional with $\mathbf{R}$ in every V'-structure. The failure of the definability property for second order logic is, again, a consequence of the second order characterization of the natural numbers by the formula φ_N (Section 2.3):[67] let $V = \{\mathbf{0}, \mathrm{s}\}$, and choose $A \subset \mathbb{N}$ which is not second order definable (there must be one, since there are only coutably many explicit definitions); let Γ consist of φ_N, all formulas $\mathbf{R}(\bar{n})$ for $n \in A$, and all formulas $\neg\mathbf{R}(\bar{n})$ for $n \notin A$. If a V-structure $\mathcal{S}$ is expanded to a model of Γ then $\mathcal{S} \models \varphi_N$, and is therefore standard, and we must have $\mathbf{R}^{\mathcal{S}} = A$.
7. The unification problem is decidable for first order logic [Robinson, 1965], but not for second order logic [Amiot, 1990].[68]
8. An important theorem of Model Theory (for first order logic), which is directly related to first order expressiveness, is Fraïssé's Theorem.

---

[65] See e.g. [Chang and Keisler, 1973; Hodges, 1993; Ebbinghaus *et al.*, 1984]).

[66] Put differently, every two models of Γ that have the same universe and same interpretation for the constants in V, must have the same interpretation for $\mathbf{R}$.

[67] The argument seems to have been part of the folklore; it can also be found in [Shapiro, 1991, Section 6.6.3]. [Shapiro, 1991] also observes that the definability property does hold for second order logic for Γ finite: let φ be the conjunction of the formulas in Γ, with $\mathbf{R}$ replaced by a variable R of the same arity, and let $\gamma[\vec{x}] \equiv_{\text{df}} \exists R(\varphi \wedge R(\vec{x}))$.

[68] The latter result is based on the undecidability of the second order term unication problem, proved in [Goldfarb, 1981].

It states, roughly, that a property of models is definable by a first order formula iff it is also recognized by a computation with a finite number of alternations between existential (nondeterministic) and universal (co-nondeterministic) guesses. A related theorem of Keisler states that a property of models is definable by a first order formula iff both it and its negation are preserved under isomorphisms and the formation of ultraproducts. Thus, any second order formula that exceeds the expressive power of first order logic, such as φ^N, defines a property of models that cannot be recognized by computations as above, and is not preserved under ultrapowers.[69]

9. First order logic satisfies a 0-1 law: if V is a vocabulary without individual constants, let n_V be the number of non-isomorphic V-structures, and, for a first order V-forumla φ, let $n_V(\varphi)$ be the number of non-isomorphic V-structures in which φ is true. Then $\mu(\varphi) =_{\mathrm{df}} \lim_{n\to\infty} \frac{n_V(\varphi)}{n_V}$ exists, and is equal to 0 or to 1.[70] Clearly, second order logic does not have this property, since there is a second order formula which is true of a finite structure iff it has an even number of elements.[71]

5.2 Lindström's and Quine's tests

Is second order logic truly a logicΓ On a technical level the answer is trivially positive. From a philosophical-ontological angle the answer is less clear.

From a model-theoretic viewpoint second order logic is merely one of many possible logics. The syntax of a logic has logical and non-logical constants. The semantics of a logic has a collection of possible interpretations (models, structures), which includes an *interpretation-dependent* assignment of meaning to the non-logical constants, and an *invariant* assignment of meaning to the logical constants. For example, in first order logic the logical constants are interpreted uniformly in all structures, whereas the vocabulary identifiers get specific interpretations in each structure. Similarly, logic with a quantifier U = 'there exist uncountably many' has U interpreted as intended in all structures. Second order logic is, then, one

[69] See e.g. [Chang and Keisler, 1973; Hodges, 1993] for detailed proofs of the Fraïssé and Keisler Theorems; Van Benthem and Doets [Benthem and Doets, 1983] have a pleasant and simplified presentation. In their original forms, common in expositions of Model Theory, these theorems are stated as characterizations of elementary equivalence.

[70] This theorem was discovered independently by Glebskii et al. [Glebskii *et al.*, 1969] and Fagin [Fagin, 1976]. It fails in the presence of constants: if φ is $R(\mathbf{c},\mathbf{c})$ (R a binary relation constant) then $\mu(\varphi) = \frac{1}{2}$. A survey of 0-1 laws for various logics is [Compton, 1988]; some more recent results can be found in [Spencer and Shelah, 1987; Kolaitis and Vardi, 1987; Kolaitis and Vardi, 1990; Kolaitis and Vardi, 1992; Spencer, 1993].

[71] However, see [Kolaitis and Vardi, 1992] for cases of second order formulas for which the 0-1 law does hold.

in a plethora of logics, in which the logical constants of first order logic are augmented by quantification over relations.

However, from a philosophical viewpoint, we might wish to reserve the term 'logic' to *a priori* concepts and truths, ones that do not depend on experience and observation. Quine [1970] suggested that the demarcation between logic and mathematics is determined by *ontological neutrality,* that is, on not assuming the existence of certain objects and structures. In particular, if the notion of *infinity* is delineated by a formalism, then that formalism is mathematical rather than logical. Quine concludes that second order logic is a mathematical theory rather than a logic.[72]

A landmark theorem of Lindström[73] states, roughly, that out of all logics, first order logic is characterized as the maximal logic that is both compact and satisfies the downward Skolem–Löwenheim property. From Quine's viewpoint, these two characteristics of first order logic are indeed litmus tests for being a logic: compactness is the failure to distinguish between the finite and the infinite, and downwards Skolem-Löwenheim is the failure to distinguish between different infinities.

Quine's argument is corroborated by the success of second order logic to capture *directly* most all mathematical practice. However, this critical view is really an afterthought. At first sight, the mere quantification over relations does not seem to be much of an ontological commitment. Metamathematically, the notion of arbitrary sets and relations underlies already the model theory of first order logic. Perhaps one might advocate quantification only over sets (unary relations) as being ontologically more prudent than quantification over relations of arbitrary arities, but this restricted variant of second order logic already begets, as we have seen, the characterization of $\mathbb{N}$. David Lewis (quoted by Hazen [1989]) has argued that even monadic third order logic is ontologically neutral, but Hazen showed that that formalism suffices to interpret all of second order logic.

Girard has suggested[74] a technical proof-theoretic criterion for a logic: a true logic must be amenable to a cut-free sequential calculus without axioms, which satisfies the subformula property (in the strict sense, see Section 4.4). The underlying intuition is that neither axioms nor rules should allow 'communication' between formulas other than by rules that explicate the logical constants in isolation. This criterion is clearly related to Quine's ontological neutrality. Indeed, even relatively weak fragments of second order logic fail Girard's criterion.[75]

---

[72] [Tharp, 1975] concurs with Quine, [Boolos, 1975] disagrees.

[73] [Lindström, 1969], see [Ebbinghaus *et al.*, 1984] for a textbook exposition.

[74] Personal communication.

[75] Nonetheless, it is possibly to recover the textual subformula property for fragments which, though logically weak, are nonetheless computationally interesting; see e.g. [Nadathur and Miller, 1990; Miller *et al.*, 1991].

5.3 Slipping from first to second order logic

In fact, the notion of 'arbitrary relation' is implicit already in first order *logic* (though not in first order languages used to describe and prove properties of *particular* first order structures). To say that a formula φ is valid is to say that it is true in all interpretations, a statement involving a universal quantification over universes as well as over the relations interpreting the predicate letters in φ. This point was made by Hilbert and Ackermann already in [1928]: *'the formalism of this [first order] calculus is clearly not a closed system.* In other words, the most basic notions of the metamathematics of first order logic are second order.'

Moreover, there is a logical construct that seems even more ontologically benign than relational quantification, and which yields nonetheless the full expressive power of second order logic, namely *partially order quantifiers.* Henkin [1961] noted that the use of quantifiers in first order formulas prevents one from expressing forms of dependence which occur naturally in both mathematics and natural language discourse [Barwise, 1979]. For instance, in the formula $\forall x \exists y \forall u \exists v\ \varphi$ there is no way to state that v depends only on u. Henkin proposed an extension of first order language with partially order quantifier-combinations, so that, for example,

$$\left.\begin{array}{l} \forall x \exists y \\ \forall u \exists v \end{array}\right\} \varphi$$

states that for all x and u, φ can be made true by suitably choosing y depending on the value of x, and choosing v depending on u. More generally, we may define a partially ordered quantifier to be a triple $Q = (\vec{x}, \vec{y}, \beta)$, where $\vec{x}, \vec{y}$ are tuples of variables, all distinct, and β is a function assigning to each variable in $\vec{y}$ a sublist of $\vec{x}$. If φ is a formula whose semantics is defined, then the semantics of $Q\varphi$ is: for all $\vec{x}$ one can find values for each variable y among $\vec{y}$, depending only on the values of the variables in $\beta(y)$ (i.e. invariant with respect to changes in values of the remaining x's), which make φ true.

At first sight it is not at all obvious that partially ordered quantifiers are ontologically less neutral than the usual nesting of first order quantifiers. However, simple partially-ordered quantifiers suffice to characterize the infinite structures (in contrast to Theorem 2.2.1). Namely, let

$$\eta \equiv_{\mathrm{df}} \exists w \begin{array}{l} \forall x\, \exists u \\ \forall y\, \exists v \end{array} (\,(\, x{=}y \leftrightarrow u{=}v\,) \wedge u \neq w\,)).$$

This formula is semantically equivalent to

$$\exists w\, \exists f,\, g\, \forall x,\, y\, (\, (x{=}y \leftrightarrow f(x){=}g(y)) \wedge f(x) \neq w\,)$$

that is to

$$\exists f\ (\ \exists g\ \forall x,\ y\ (x{=}y \leftrightarrow f(x){=}g(y)) \wedge \exists w\ \forall x\ f(x) \neq x\ ).$$

Since the first conjunct is equivalent to $\mathit{Inj}[f]$ and the second conjunct is equivalent to $\neg \mathit{Surj}[f]$, it follows that η is true exactly in the infinite structures.[76]

5.4 Higher order logic as mathematics: Henkin's semantics

Quine's position, that second order logic is a mathematical rather than a logical formalism, justifies a model theory in which quantification over relations (and functions) is a mathematical, rather than logical operation. That is, the interpretation of higher order quantification is put on an equal footing with the vocabulary constants, and is part of the specification of a model rather than an invariant through all models. This semantics, which we now describe in more detail, is due to Henkin.[77]

Let V be a vocabulary. A **Henkin-V-prestructure** $\mathcal{H}$ (of finite order logic) consists of

- a non-empty universe A;
- an interpretation in A of the V-constants; and
- for each type τ, a collection D^τ, where $D^\iota = A$, and $D^{(\tau_1 \cdots \tau_k)} \subseteq \mathcal{P}(D^{\tau_1} \times \cdots \times D^{\tau_k})$.

Assignments η into $\mathcal{H}$ are defined as in the tarskian semantics (Section 3.1 above), except that if R is a variable of type τ, then we require that $\eta(R) \in D^\tau$. Semantic satisfaction of formulas, $\mathcal{H}, \eta \models \varphi$, is then defined inductively, as usual. Thus, tarskian models of finite order logic may be regarded as special Henkin interpretations, the *'canonical'* interpretations, where $D^\tau = A_\tau$ for all types τ.

A **Henkin-structure** of finite order logic is a Henkin-prestructure $\mathcal{H}$ that is closed under definability: for each formula φ, assignment η into $\mathcal{H}$, and type $\tau = (\tau_1 \ldots \tau_k)$, [78]

---

[76] The formula η is due to Ehrenfeucht (reported in [Henkin, 1961]). [Enderton, 1970] and [Walkoe, 1970] independently showed that every Σ^1_1 formula is semantically equivalent to a formula in the language L^H_1 of first order logic augmented with partially-ordered quantifiers. Enderton also showed that every formula of L^H_1 is semantically equivalent to a Δ^1_2 formula. Harel [1979] showed that for a natural modification of the semantics of partially-order quantifiers L^H_1 is expressively equivalent to full second order logic, and M. Motowski showed that expressive equivalence with full second order logic can be obtained, alternatively, by closing the collection of partially-ordered quantifiers under a duality operation. For detailed surveys on partially-ordered quantifiers see [Mundici, 1985] and [Krynicki and Mostowski, 1994].

[77] See [Henkin, 1950]. A correction to Henkin's treatment is in [Andrews, 1972a].

[78] $[a_1 \ldots a_k / x_1 \ldots x_k]\eta$ is the assignment that differs from η only in assigning $a_1 \ldots a_k$ to $x_1 \ldots x_k$, respectively.

$$\{\vec{a} \in A_\tau \mid \mathcal{H}, [a_1 \ldots a_k/x_1 \ldots x_k]\eta \models \varphi\} \in D^\tau.$$

A *Henkin-V-prestructure for type* σ is a Henkin-V-prestructure with D^τ given for the subtypes τ of σ (including σ itself). Henkin-V-structures for type σ are defined analogously.

An even more squarely mathematical interpretation of higher order quantification arises from viewing finite order logic as a syntactic variant of a **first order theory of types,** TT. We present this theory first for *monadic second* order formulas, i.e. formulas with only unary relations. Given a vocabulary V, let V^{S} be an extension of V with the relation-constants T_ι, $T_{(\iota)}$, and E, of arities 1,1 and 2, respectively. $T_\iota(x)$ is intended to state 'x is an individual', $T_{(\iota)}(x)$ — 'x is a set', and $E(x,y)$ — 'x is an element of y.'

A Henkin-V-prestructure $\mathcal{H}$ for (ι) determines a unique V^{S}-structure $\mathcal{S} = \mathcal{S}(\mathcal{H})$, as follows.

- The universe of $\mathcal{S}$ is $A \cup D^{(\iota)}$ where A and $D^{(\iota)}$ are the universes of $\mathcal{H}$ for types ι and (ι);
- the interpretations in $\mathcal{S}$ of the V-constants is the same as their interpretation in $\mathcal{H}$, with functions (which in $\mathcal{H}$ are defined only over A) extended to arguments in $D^{(\iota)}$ arbitrarily;
- T_ι is interpreted as A, $T_{(\iota)}$ as $D^{(\iota)}$, and E as $\in$.

Each monadic V-formula φ can be rephrased as a first order V^{S}-formula φ^{S}, where φ^{S} is obtained from φ by (1) replacing each atom $R(t)$ (R a variable) by $E(t,R)$; and (2) relativizing quantifiers over individuals to T_ι, and quantifiers over sets to $T_{(\iota)}$.[79] In φ^{S} all variables are understood as object-variables.

It is then easy to prove:

Lemma 5.4.1. *A formula φ is true in $\mathcal{H}$ iff φ^{S} is true in $\mathcal{S}(\mathcal{H})$.*

Thus, if φ^{S} is valid, then φ is true in all Henkin-prestructures. The converse is not generally true: for example, $\exists x\, x{=}x$ is true in all Henkin-prestructures (because the universe of a structure is non-empty), but $\exists x\, (T_\iota(x) \wedge (x{=}x))$ is not valid. More generally, not every V^{T}-structure $\mathcal{S}$ is $\mathcal{S}(\mathcal{H})$ for some $\mathcal{H}$. Indeed, the structures $\mathcal{S}(\mathcal{H})$ have the following properties (compare [Benthem and Doets, 1983]):

[79] That is, φ^{S} is defined by recurrence on formulas as follows. $(R(t))^{\mathrm{S}} = E(t,R)$ (R a set-variable), $\alpha^{\mathrm{S}} = \alpha$ for other atomic formulas, $(\neg\varphi)^{\mathrm{S}} = \neg(\varphi^{\mathrm{S}})$, $(\varphi \star \psi)^{\mathrm{S}} = (\varphi^{\mathrm{S}} \star \psi^{\mathrm{S}})$ for binary connectives $\star$, $(\forall x\varphi)^{\mathrm{S}} = \forall x(T_\iota(x) \to \varphi^{\mathrm{S}})$, $(\exists x\varphi)^{\mathrm{S}} = \exists x\,(T_\iota(x) \wedge \varphi^{\mathrm{S}})$, $(\forall R\varphi)^{\mathrm{S}} = \forall R(T_{(\iota)}(R) \to \varphi^{\mathrm{S}})$, $(\exists R\varphi)^{\mathrm{S}} = \exists R(T_{(\iota)}(R) \wedge \varphi^{\mathrm{S}})$.

1. V-correctness: $T_\iota(\mathbf{c})$ for object-constants $\mathbf{c}$ of V, $\mathbf{R}(x_1 \ldots x_k) \to \bigwedge_{i=1}^k T_\iota(x_i)$ for (k-ary) relation-constants $\mathbf{R}$ of V, and $\bigwedge_{i=1}^k T_\iota(x_i) \to T_\iota(\mathbf{f}(\vec{x}))$ for (k-ary) function-constants $\mathbf{f}$ of V;
2. non-emptiness: $\exists x\ T_\iota(x)$;
3. disjointness: $T_\iota(x) \to \neg T_{(\iota)}(x)$;
4. inclusion: $T_\iota(x) \vee T_{(\iota)}(x)$;
5. elementhood: $E(x,y) \to T_\iota(x) \wedge T_{(\iota)}(y)$
6. extensionality: $T_{(\iota)}(x) \wedge T_{(\iota)}(y) \wedge \forall z\,(E(z,x) \leftrightarrow E(z,y)) \to x = y$.

Lemma 5.4.2. *If $\mathcal{S}$ has properties 1–6 above, then $\mathcal{S} = \mathcal{S}(\mathcal{H})$ for some Henkin-V-prestructure for $\mathcal{H}$ type (ι).*

Proof. Given $\mathcal{S}$, with universe $|\mathcal{S}|$, let $\mathcal{H}$ have $D^\iota =_{\mathrm{df}} (T_\iota)^{\mathcal{S}}$ as universe of individuals, let the interpretation of the V-constants be the same as in $\mathcal{S}$ (with the functions restricted to D^ι), and let the universe $D^{(\iota)}$ of sets consist of the sets $\{a \in D^\iota \mid E^{\mathcal{S}}(a,s)\}$ for $s \in (T_{(\iota)})^{\mathcal{S}}$. By the V-correctness and non-emptiness properties of $\mathcal{S}$, $\mathcal{H}$ is a Henkin-V-prestructure.

We claim that $\mathcal{S}(\mathcal{H})$ is isomorphic to $\mathcal{S}$. Let a function $j : |\mathcal{S}| \to |\mathcal{S}(\mathcal{H})|$ be defined by

$$j(x) = \begin{cases} x & \text{if } x \in (T_\iota)^{\mathcal{S}} \\ \{a \in I \mid E^{\mathcal{S}}(a,x)\} & \text{if } x \in (T_{(\iota)})^{\mathcal{S}} \end{cases}$$

Then j is well-defined by the disjointness and inclusion conditions, it is surjective by definition of $\mathcal{S}(\mathcal{H})$, and injective by the extensionality condition. j preserves the V-constants, T_ι, and $T_{(\iota)}$ by definition, and E by the elementhood condition. ∎

Let Θ_2 be the set of formulas listed under conditions 1–6 above. We then have:

Lemma 5.4.3. *A monadic formula φ is true in all Henkin-prestructures iff $\Theta_2 \models \varphi^{\mathrm{S}}$.*

Proof. Suppose that φ is true in all Henkin-prestructures. Towards showing $\Theta_2 \models \varphi^{\mathrm{S}}$ assume $\mathcal{S} \models \Theta_2$. Then, by Lemma 5.4.2, $\mathcal{S} = \mathcal{S}(\mathcal{H})$ for some Henkin-prestructure $\mathcal{H}$. By assumption we have $\mathcal{H} \models \varphi$, and so $\mathcal{S}(\mathcal{H}) \models \varphi^{\mathrm{S}}$ (by Lemma 5.4.1), whence $\mathcal{S} \models \varphi^{\mathrm{S}}$.

Conversely, suppose that $\Theta_2 \models \varphi^{\mathrm{S}}$, and let $\mathcal{H}$ be a Henkin-prestructure. Then $\mathcal{S}(\mathcal{H}) \models \Theta_2$, and so $\mathcal{S}(\mathcal{H}) \models \varphi^{\mathrm{S}}$, whence, by Lemma 5.4.1, $\mathcal{H} \models \varphi$. ∎

A similar duality exists for provability:

Lemma 5.4.4. *A monadic V-formula φ is provable in L_2 iff φ^{S} is provable in first order logic from formulas of the form*

$$\exists R\ \forall x\ (\ E(x,R) \leftrightarrow \psi\ ) \qquad (*)$$

(where R is not free in ψ).

The proof is straightforward by induction on (the length of) derivations.

Combining Lemmas 5.4.3 and 5.4.4 with the Completeness Theorem for first order logic we obtain:

Theorem 5.4.5. [Henkin]

1. *A monadic second order formula φ is true in all Henkin-prestructures iff φ^{S} is provable in first order logic from Θ_2.*
2. *φ is true in all Henkin-structures iff φ^{S} is provable in first order logic from Θ_2 and all formulas of the form (*).*

The generalization of Theorem 5.4.5 from monadic to full second order logic is straightforward.

5.5 Henkin completeness for full finite order logic

We now outline a more general duality, between full finite order logic and a first order theory. Given a vocabulary V, let V^{T} be the vocabulary with: (1) the constants of V; (2) for each type τ a unary relation T_τ; and (3) for each type $\tau = (\tau_1 \dots \tau_k)$ a $k+1$-ary relation E_τ. To each finite order V-formula φ we correspond a first order V^{T}-formula φ^{T}, as follows. (1) replace each atomic formula $R(t_1 \dots t_k)$ in φ, where R is a variable of type $\tau = (\tau_1 \dots \tau_k)$, by $E_\tau(t_1 \dots t_k, R)$; and (2) for each type τ, relativize quantifiers over τ to T_τ.[80]

Like for monadic second order formulas, a (full) Henkin-structure $\mathcal{H}$ for V determines a unique V^{T}-structure $\mathcal{S}(\mathcal{H})$. Each such structure satisfies the following set Θ_ω of formulas, analogous to Θ_2 (except for the inclusion condition, which cannot be stated because there are infinitely many types): V-correctness; non-emptiness; disjointness: $T_\tau(x) \to \neg T_\sigma(x)$ for distinct τ and σ; elementhood: for $\tau = (\tau_1 \dots \tau_k)$, $E_\tau(x_1 \dots x_k, y) \to T_\tau(y) \wedge T_{\tau_1}(x_1) \wedge \dots \wedge T_{\tau_k}(x_k)$; and extensionality: for $\tau = (\tau_1 \dots \tau_k)$, $T_\tau(x) \wedge T_\tau(y) \wedge \forall \vec{z}\,(E_\tau(\vec{z}, x) \leftrightarrow E_\tau(\vec{z}, y)) \to x = y$.

For a Henkin-prestructure $\mathcal{H}$ let $\mathcal{S}(\mathcal{H})$ be a V^{T}-structure defined analogously to the definition above for the case where $\mathcal{H}$ is a prestructure for (ι) only. Analogously to Lemma 5.4.4 we have for L_ω:

Lemma 5.5.1. *A V-formula φ is provable in L_ω iff φ^{T} is provable in first order logic from formulas of the form*

$$\exists R\, \forall \vec{x}\, (\, E_{(\tau_1 \dots \tau_k)}(\vec{x}, R) \leftrightarrow \psi\,) \qquad (R \text{ not free in } \psi) \qquad (**)$$

[80] That is, $(R^{(\tau_1 \dots \tau_k)}(t_1 \dots t_k))^{\mathrm{T}} = E_{(\tau_1 \dots \tau_k)}(t_1 \dots t_k, R)$ (R a variable of type $(\tau_1 \dots \tau_k)$); $\alpha^{\mathrm{T}} = \alpha$ for other atomic formulas; $(\neg\varphi)^{\mathrm{T}} = \neg(\varphi^{\mathrm{T}})$; $(\varphi \star \psi)^{\mathrm{T}} = (\varphi^{\mathrm{T}} \star \psi^{\mathrm{T}})$ for binary connectives $\star$; $(\forall x \varphi)^{\mathrm{T}} = \forall x (T_\iota(x) \to \varphi^{\mathrm{T}})$, $(\exists x \varphi)^{\mathrm{T}} = \exists x (T_\iota(x) \wedge \varphi^{\mathrm{T}})$, $(\forall R^\tau \varphi)^{\mathrm{T}} = \forall R (T_\tau(R) \to \varphi^{\mathrm{T}})$, $(\exists R^\tau \varphi)^{\mathrm{T}} = \exists R (T_\tau(R) \wedge \varphi^{\mathrm{T}})$.

Theorem 5.5.2. [Henkin]

1. *A finite order formula φ is true in all Henkin-prestructures iff φ^{T} is provable in first order logic from Θ_ω.*
2. *Consequently, a finite order formula φ is true in all Henkin-structures iff φ^{T} is provable in first order logic from Θ_ω and formulas of the form (**).*

Proof. We prove (1). Suppose $\Theta_\omega \vdash \varphi^{\mathrm{T}}$, and let $\mathcal{H}$ be a Henkin–prestructure. Then $\mathcal{S}(\mathcal{H}) \models \Theta_\omega$, and therefore $\mathcal{S}(\mathcal{H}) \models \varphi^{\mathrm{T}}$. As in Lemma 5.4.1, this implies that $\mathcal{H} \models \varphi$.

Conversely, suppose that φ is true in all Henkin-prestructures. Towards showing that $\Theta_\omega \models \varphi^{\mathrm{T}}$, let $\mathcal{S}$ be a model of Θ_ω, and let $\mathcal{S}'$ be its substructure generated by the elements of $\bigcup_\tau (T_\tau)^{\mathcal{S}}$. Then, analogously to Lemma 5.4.2, $\mathcal{S}'$ is isomorphic to $\mathcal{S}(\mathcal{H})$ for some Henkin-prestructure $\mathcal{H}$. Since φ is assumed true in $\mathcal{H}$, it follows, as in Lemma 5.4.1, that φ^{T} is true in $\mathcal{S}(\mathcal{H})$, and therefore in $\mathcal{S}'$. But each quantifier in φ^{T} is relativized to some T_τ, so φ^{T} is true also in $\mathcal{S}$. We have shown that $\Theta_\omega \models \varphi^{\mathrm{T}}$, and so, by the Completeness Theorem, $\Theta_\omega \vdash \varphi^{\mathrm{T}}$. ∎

The soundness of L_2 for Henkin's semantics can be used to establish independence results. For instance,

Theorem 5.5.3. *The principle of choice is not provable in L_2.*

Proof. If Choice were a theorem of L_2, then every Henkin-structure would satisfy choice. Consider the Henkin-structure $\mathcal{S}$ for the empty vocabulary, with a three element universe $\{a, b, c\}$, and where the relations are exactly the definable ones.[81] Call valuations η and η' isomorphic if: (1) for all individual variables x, y, $\eta(x) = \eta(y)$ iff $\eta'(x) = \eta'(y)$; and (2) for all r-ary relation variable R and individual variables $x_1 \ldots x_r$, $\langle \eta(x_1) \ldots \eta(x_r) \rangle \in \eta(R)$ iff $\langle \eta'(x_1) \ldots \eta'(x_r) \rangle \in \eta'(R)$. It is easy to see, by induction on φ, that if η and η' are isomorphic then $\mathcal{S}, \eta \models \varphi$ iff $\mathcal{S}, \eta' \models \varphi$.

In $\mathcal{S}$ we have $\forall x \exists y\, y \neq x$, which by Choice implies

$$\mathcal{S} \models \exists R \forall x \exists! y\, R(x, y) \wedge y \neq x.$$

If Choice were provable then, by the definition of $\mathcal{S}$, this implies that for some formula $\varphi[x, y]$ we have $\mathcal{S} \models \forall x \exists! y\, (\varphi[x, y] \wedge y \neq x)$. In particular, there is a valuation η that assigns a to all individual variables in φ other than y, and $\emptyset$ to all relation variables, and such that $\mathcal{S}, \eta \models \varphi$. Say $\eta(y) = b$. By the observation above we also have $\mathcal{S}, [c/y]\eta \models \varphi$, contradicting $\mathcal{S}, \eta \models \forall x \exists! y\, \varphi[x, y]$.[82] ∎

[81] That is, definable by second order formulas with equality.

[82] A more interesting counter-example: consider the Henkin prestructure that consists of the standard model of Peano's Arithmetic with the first order definable relations.

5.6 Finite order logic as a second order theory

The construction above reduces finite order logic to a first order theory, by allowing non-canonical semantic interpretations. We can also reduce finite order logic to a *second*-order theory, *without* allowing non-canonical interpretations. For each type $\tau = (\tau_1 \dots \tau_k)$, consider the following formula, expressing the representability of every relation over $T_{\tau_1} \dots T_{\tau_k}$ by an object in T_τ:

$$Rep_\tau \quad \equiv_{\mathrm{df}} \quad \forall R\, \exists s\, (\, T_\tau(s) \,\wedge\, \forall y_1 \dots y_k .\, \bigwedge_{i=1}^{k} T_{\tau_i}(y_i) \;\rightarrow\; (\, R(\vec{y}) \leftrightarrow E_\tau(\vec{y}, s)\,)\,)$$

To each canonical Henkin-V-prestructure $\mathcal{H}$ corresponds a V^{T}-structure $\mathcal{R}(\mathcal{H})$ which satisfies Rep_τ for every type τ, as well as Θ_ω. Conversely, if $\mathcal{S}$ is a V^{T}-structure that satisfies Θ_ω and Rep_τ for every type τ, then the substructure $\mathcal{S}'$ (as defined above) of $\mathcal{S}$ is isomorphic to $\mathcal{R}(\mathcal{H})$ for some canonical $\mathcal{H}$. We therefore obtain:[83]

Theorem 5.6.1. *A formula φ of finite order logic is valid (in the standard sense) iff the second order formula φ^{T} is true in every model of $\Theta_\omega \cup \{Rep_\tau\}_{\tau \text{ a type}}$.*

For each formula φ, the proof of Theorem 5.6.1 uses only formulas in Θ_ω that involve types in φ. Let χ_φ be the conjunction of all these formulas. Note that χ_φ is a Π^1_1 formula. We have thus obtained:

Theorem 5.6.2. *A formula φ of finite order logic is valid iff the Σ^1_1 formula $\chi_\varphi \rightarrow \varphi^{\mathrm{T}}$ is valid.*

Van Benthem and Doets [1983, Section 4.3] further refine this result: they show how χ_φ can be replaced by a *monadic* Π^1_1 formula. Montague [1965b] pointed out that Theorem 5.6.2 can be further extended to transfinite order logic with orders that are ordinals describable in finite order logic.

Note that Theorem 5.6.2 refers to validity in *all* structures. When a particular structure is considered, for example $\mathcal{N}$, then 3.7.1 shows that

---

Choice over arithmetic relations sometimes defines (hyperarithmetical) non-arithmetical relations (see e.g. [Rogers, 1967]), so choice fails in this Henkin-structure. [Andrews, 1972b] shows that choice for any given type is not a theorem of full finite order logic, even if the latter is augmented by the Axiom Schema of Description. The Axiom of Description for type α states that there is a functional that for every singleton set X of α's as input returns the sole element of X.

[83] A preliminary form of this theorem seems to be due to Hintikka [1955], which Montague generalized in [1965b]. Shapiro [1991] cites [Montague, 1965b] as an independent source, stating that Montague attributes the result to David Kaplan. Expositions are also given in [Kreisel and Krivine, 1964; Benthem and Doets, 1983; Shapiro, 1991].

even validity in $\mathcal{N}$ of Π^1_1 formulas is *not* reduced to validity in $\mathcal{N}$ of Σ^1_1 formulas. Theorem 5.6.2 states that the truth in $\mathcal{N}$ of a finite type formula ψ, i.e. the validity of the formula $\varphi_N \wedge D_{+,\times} \to \psi$ (see Section 2.3), is reducible to the validity (in all structures) of a Σ^1_1 formula of a totally different nature, namely one that refers explicitly to the coding of a type structure.

A striking consequence of Theorem 5.6.2 is:

Theorem 5.6.3. *The set of (numeric codes of) valid second order formulas is not definable over* $\mathbb{N}$ *in finite order logic.*[84]

Proof. Suppose $\sigma[x]$ is a finite-order formula in the vocabulary of arithmetic, such that $\sigma[\bar{n}]$ is true (in the standard model) iff n codes a valid second order formula. Let κ be a primitive recursive function that maps the numeric code of a formula φ of finite order logic to the code of $\chi_\varphi \to \varphi^{\mathrm{T}}$. Then, for each formula φ of finite order logic, φ is valid iff $\chi_\varphi \to \varphi^T$ is valid, i.e. iff $\sigma[\overline{\kappa\#\varphi}]$ is true in the standard model. In particular, if φ is a finite order formula in the vocabulary of arithmetic, then φ is true in the standard model iff $\nu \to \varphi$ is valid, i.e. iff $\varphi[\kappa\#(\nu \to \varphi)]$ is true in the standard model. Thus, the finite order formula

$$\tau[x] \equiv_{\mathrm{df}} D \to \sigma[\kappa(\mathit{impl}(\#\nu, x))],$$

where D is the primitive recursive definition of *impl* and of κ, is a truth definition over $\mathbb{N}$ for all finite order formulas, contradicting Tarski's Theorem 2.6.1. ∎

Thus, while the set of valid second order formulas *is* definable by the *validity* of a suitable third order formula (Section 3.7), it is not definable if relation variables are restricted to numeric relations. This contrast is not surprising, since second order formulas go a long way in defining higher cardinals (2.2).

6 Restricted higher order logic

Higher order logics have a useful expressive power, but at the price of technical difficulties and a problematic ontology. This has motivated restrictions of higher order logic, where each restriction attempts to preserve

[84] By Montague's observation mentioned above, that set is not definable even in transfinite order number theory, as long as the order-ordinals are all definable in finite order logic. It is easy to see that second order truth is definable in set theory by a Π_2 formula that renders $(\forall$ structures $\mathcal{S})$ $(\mathcal{S} \models \varphi)$ (see e.g. [Boolos, 1975]). This is not a definition over $\mathbb{N}$, since the set quantifiers range here over arbitrary sets in the intended hierarchy of sets; however, using the Levy–Montague Reflection Principle [Levy, 1960], there is an ordinal α such that all statements above for the (countably many) second order formulas φ are true exactly when they are true in V_α. Then second order truth is definable over $\mathbb{N}$ in α-order logic.

certain forms of expressiveness, while eliminating or reducing technical or ontological problems. These restrictions might be classified into restrictions on the *expressiveness*, the *semantics*, and the *proof theory*, though these three aspects are often intertwined.

6.1 Restricted expressiveness 1: Monadic second order logic

Two natural language restrictions are monadic second order logic and fixpoint logic. The former is driven by syntactic form, the latter by semantics. In **monadic second order logic** the only higher order variables are ones ranging over monadic relations, i.e. sets.

Monadic first order logic is decidable [Löwenheim, 1915], and, in the absence of functions, so is monadic second order logic [Skolem, 1919; Behmann, 1922]. However, with functions present monadic second order logic is quite complex: the definition φ_N above of $\mathbb{N}$ is monadic (Section 2.3). Primitive-recursive pairing and projection functions are axiomatized by their recursion equations, so it follows that the truth of a second order formula over the standard structure of the natural numbers can be expressed by a monadic second order formula.[85] Thus, monadic second order logic is not compact, possesses neither the Upward Skolem-Löwenheim Property nor the Beth Definability Property, and the set of monadic second order formulas valid in all structure is not analytical, let alone effectively enumerable.[86]

Monadic second order logic also fails to satisfy the Downward Skolem–Löwenheim property. Our proof above that downward Skolem–Löwenheim is not satisfied by full second order logic relies on using quantified binary-relations to define uncountable structures. However, it is easy to define in monadic second order logic particular classes of uncountable structures, such as dense linear orderings with endpoints that contain their Dedekind cuts.[87] From this the failure of the Downward Skolem–Löwenheim property

[85] For example, a formula $\varphi \equiv \forall R \psi$, where R is binary and ψ is first order, is expressed by $\forall X \chi$, where χ arises from ψ by replacing each subformula $R(t, t')$ by $X(p(t, t'))$, where p is the identifier for a primitive-recursive pairing function. That is, the truth of φ over the natural numbers is expressed by the monadic second order formula

$$D \rightarrow (\forall X \subseteq N)(\chi^N),$$

where D is the primitive recursive definition of the functions used in ψ as well as for p, and where χ^N is χ relativized to N.

[86] Moreover, monadic second order logic does not have the Craig interpolation property [Mostowski, 1968], which both first order and full second order logic have. A logic L *has the interpolation property* if for every valid L-sentence $\varphi \rightarrow \psi$ there is a sentence χ, which uses only non-logical constants that occur in both φ and ψ, and such that $\varphi \rightarrow \chi$ and $\chi \rightarrow \psi$ are valid. A proof of the interpolation property for first order logic can be found in most logic textbooks, see e.g. [Ebbinghaus *et al.*, 1984]. For second order logic the property is trivial (see e.g. [Shapiro, 1991]): let $R_1 \cdots R_k$ be the vocabulary relation identifiers that occur in φ but not in ψ, and let χ be $\exists R_1 \cdots \exists R_k \varphi$.

[87] Let φ be $\psi \wedge \chi$, where ψ is the first order formula axiomatizing dense linear orderings,

follows as before.

In view of the similarities above between monadic and full second order logics, one might wonder about the point in restricting predicates to monadic ones. It turns out, however, that for many *theories*, this restriction is crucial, and leads to interesting results. Notably, the full second order theory of zero and a successor function (*without* additional function symbols) is essentially second order arithmetic (since the graphs of all primitive-recursive functions are definable), but the *monadic* second order theory of even *two* successors (i.e. the monadic second order theory of binary trees) is decidable [Rabin, 1969]. A discussion and compendium of monadic second order theories can be found in [Gurevich, 1985].

6.2 Restricted expressiveness 2: Fixpoint logics

Because inductive definitions play particularly important roles in various applications of higher order logic, it is natural to isolate them, and to consider extensions of first order logic with fixpoint constructs.[88]

The study of proof calculi for first order logic extended with fixpoints is implicit in the development of the theory of inductive definitions over arbitrary structures [Moschovakis, 1974]. However, syntactic logical formalisms that incorporate explicitly a fixpoint operator seem to have originated independently in two areas of Theoretical Computer Science: logics of programs and database theory. Syntactic logical formalisms that explicitly incorporate a fixpoint operator were first introduced in relation to programming language semantics [Scott and de Bakker, 1969; Park, 1970; Hitchcock and Park, 1973]. Consequently, the fruitful investigations of propositional modal formalisms for reasoning about programs led to the

---

and $\chi \equiv_{\mathrm{df}} \forall X \forall Y \ (X < Y \rightarrow \exists z \ (X \leq z \wedge z \leq Y))$, where $X < Y \equiv_{\mathrm{df}} \forall x \in X \, \forall y \in Y \ x < y$, $X \leq z \equiv_{\mathrm{df}} \forall x \in X \ x \leq z$, and $z \leq Y \equiv_{\mathrm{df}} \forall y \in Y \ z \leq y$. Then φ has (infinite) models, e.g. $[0,1]$ with the standard order. But φ has no countable model, because a countable model of ψ must be isomorphic to the rationals, with or without one or both endpoints (Cantor's Countable Order Theorem, see e.g. [Ebbinghaus *et al.*, 1984]), and χ is false for these structures.

[88] Adding fixpoint constructs to logic is somewhat different from formal *theories* for inductive definitions and iterated inductive definitions, which are usually formulated over arithmetic or analysis. These theories were initiated in [Kreisel, 1963], and studied extensively in the 1960's and 1970's; see e.g. [Buchholz *et al.*, 1981]. If n is a natural number, and $A \subseteq \mathbb{N}$, then the collection ID_n of sets ***definable by n-fold iterated inductive definitions from*** A, is defined by: $ID_0 =_{\mathrm{df}} \emptyset$; $ID_{n+1} =_{\mathrm{df}}$ the collection of fixpoints of operators $\lambda P.\varphi$, where φ is a first-order formula with sets in ID_n as parameters, and P is positive in φ. More generally, if $\prec$ is a fixed well-founded ordering, with 0 as least element, then the collections ID_α, for α represented within $\prec$, are defined using a uniform form of the definition above, so that it can be iterated into the transfinite without recourse to infinitely many operators on the way. Namely, each first order formula $\varphi[P,Q]$ (of the language of arithmetic extended with the two unary predicates P,Q), with P positive and two free variables x, a, generates a transfinite sequence φ_a (a in the field of $\prec$) of sets, as follows. $\varphi_0 =_{\mathrm{df}} \emptyset$; $\varphi_a =_{\mathrm{df}} \mu P.\lambda x.\, \varphi[P, \varphi_{\prec a}, x, a]$, where $\varphi_{\prec a} =_{\mathrm{df}} \varphi_b$ if a is the successor of b in $\prec$, and $\varphi_{\prec a} =_{\mathrm{df}} \cup_{n \prec a} \{n\} \times \varphi_n$ if a is a limit point of $\prec$.

study of the propositional μ-calculus, that is, propositional logic enriched with a fixpoint construct [Kozen, 1983].[89]

In database theory the interest in extending first order logic with a fixpoint operator [Chandra and Harel, 1982] evolved from the expressive limitations of first order logic, e.g. the impossibility to define (uniformly over all structures) the transitive closure of a binary relation, even over finite structures [Aho and Ullman, 1979].

Several fixpoint operators are possible of which the most important is fixpoint for monotone first order operators; that is, one considers an extension $FO\mu$ of the language of first order logic (over a vocabulary V) with the clause: if φ is a formula in which all occurrences of the k-ary relational identifier R are positive, and $\vec{t} = (t_1 \ldots t_k)$ is a tuple of terms, then $(\mu R.\lambda x_1 \ldots x_k \varphi)(\vec{t})$ is a formula, intended to denote the least fixpoint of the monotone operator $R \mapsto \lambda \vec{x}.\, \varphi[R]$ (see Section 2.7). Note that φ may be a formula of $FO\mu$, not necessarily a first order formula. As noted in Section 2.7, the positive formulas yield a natural, syntactically recognized, collection of monotone operators, which are guaranteed to have a least fixed point.[90]

One should be careful, however, in specifying the formulas to which μ is applied here. Under a weak reading, R is said to be positive in φ if it is not in the negative scope of any implication or negation. Under a stronger reading, R is, positive if it is in the negative scope of negation or implication an even number of times. If φ is first order, then the two readings are equivalent, because the two readings are identical for formulas in prenex-disjunctive form. However, if φ uses μ as in $\mu R.\lambda x.\ \neg \mu S.\lambda y.\, \varphi[R, S]$, (where R is negative[91] in φ and S is positive in φ), then the interaction between the two fixpoints may get to be too complex to be rendered with the weaker reading.[92]

It is easy to see that the truth of number theoretic formulas is reducible

---

[89] A finite model theorem and a decision procedure for the propositional μ-calculus are proved in [Kozen, 1988; Kozen and Parikh, 1983], and a 0-1 law in [Blass *et al.*, 1985]. A recent study on the scope of expressiveness of the μ-calculus is [Lubarsky, 1993].

[90] Other fixpoint operators include inflationary fixpoint [Gurevich and Shelah, 1986; Leivant, 1990c] and existential fixpoints [Blass and Gurevich, 1987]. Additional variants are mentioned in Section 8.4 below, in relation to computational complexity. Dual notions, of co-induction and largest fixpoints, have emerged recently in programming language theory as important in dealing with infinite objects, such as streams [Aczel and Mendler, 1989; Milner and Tofte, 1991; Pitts, 1993].

[91] that is, in the negative scope of an odd number of implications and negations

[92] However, the two readings are equivalent over finite structures, as proved by Immerman [1986], leading to a normal form theorem: every formula is equivalent to one with a single occurrence of μ. The first version is the one studied in [Moschovakis, 1974]. An attractive related language, Lower Predicate Calculus with Reflection (LPCR), is presented in [Moschovakis, 1993]. A completeness theorem for an interesting fragment of the language is in [Barwise and Moschovakis, 1978], a result improved recently by Katherine St. John (1993, in preparation).

to validity of $FO\mu$ formulas. Let φ_Z be defined like φ_N in Section 2.3, except that N is replaced by

$$Z \equiv_{\mathrm{df}} \mu R\lambda x.\, x{=}\mathbf{0} \vee \exists y.\, (x{=}\mathrm{s}y \wedge R(y)).$$

Then a formula ψ in the language of Peano's Arithmetic is true iff the formula $\varphi_Z \wedge D_{+,\times} \to \psi^Z$ is valid.

From this it immediately follows that the set of valid $FO\mu$ formulas is not arithmetical (under canonical coding). [93] Consequently, there is no formal system that generates exactly the valid $FO\mu$ formulas. Also, since the standard model of arithmetic is definable in $FO\mu$, it follows that $FO\mu$ fails to be compact or to have the upward Skolem-Löwenheim property. However, it does have the Downwards Skolem-Löwenheim property.[94]

While a complete deductive system for fixpoint logic is impossible, natural formalisms do exist for it. Two salient properties of $\mu R.\varphi$ that are incorporated in every such formal calculus are:

1. Closure under $\lambda R\lambda\vec{x}.\varphi$, i.e.

$$\forall\vec{x}.[\mu R.\varphi/R]\varphi \to (\mu R.\lambda\vec{x}\,\varphi)(\vec{x}).$$

   Or, as an inference rule,

$$\frac{[\mu R.\varphi/R]\varphi(\vec{t})}{(\mu R.\lambda\vec{x}\,\varphi)(\vec{t})}.$$

2. Minimality property: for every relation Q of the proper arity, if $[Q/R]\varphi \subseteq Q$, then $\mu R\lambda\vec{x}\varphi \subseteq Q$. This can be rendered in first order logic as a schema, allowing the substitution of any first order definable relation $\lambda\vec{z}\psi$ for Q above:

$$\forall\vec{x}\,(\,[\lambda\vec{z}\psi/R]\varphi \to [\vec{x}/\vec{z}]\psi\,) \;\to\; \forall\vec{z}\,(\,(\mu R\lambda\vec{x}.\,\varphi)(\vec{z}) \to \psi\,)$$

6.3 Restricted semantics: Weak second order logic

An important semantically restricted higher order logic is *weak* second order logic, with relation-variables ranging over *finite* relations, and with no function variables. That is, a formula φ in the relational variant of second order logic is **f-true** in a structure $\mathcal{S}$ (notation: $\mathcal{S} \models^f \varphi$) if it is true in $\mathcal{S}$ with the relation-variables ranging over finite relations.

The second order definition of the natural numbers can be amended to weak second order logic as follows. Let $\Gamma^d =_{\mathrm{df}} \{\forall x\, \mathrm{s}^{[n]}(x) \neq x\}_{n\geq 1}$, and define

[93] In fact, the set of valid $FO\mu$ formulas seems to be complete Π^1_2, as shown in privately communicated draft proofs by Y. Moschovakis and by K. Doets.

[94] Privately communicated draft proofs by Y. Moschovakis and K. Doets.

$$N'[x] \equiv_{\mathrm{df}} \exists R\,(\,R(x) \wedge \forall u\,(\,R(u) \rightarrow u = \mathbf{0} \vee \exists v\,R(v) \wedge u = \mathbf{s}(v)\,)\,)$$

In models of Γ^d the property N', with R ranging over finite sets, is true of an element a iff a is the denotation of a numeral. Therefore the formula

$$\varphi'_N \quad \equiv_{\mathrm{df}} \quad \varphi_{\mathrm{suc}} \wedge \forall x\,N'[x]$$

defines the natural numbers up to isomorphism in every model of Γ^d.[95]

Primitive recursive functions can be referred to and defined via second order definitions of their graphs.[96] It follows that to each number theoretic formula φ there corresponds a second order interpretation φ' of φ, so that φ is true in $\mathbb{N}$ iff $\Gamma^d \models \varphi'$ in weak second order logic. Therefore, the decision problem for f-validity is not in the arithmetical hierarchy, let alone recursively enumerable.

Also, since weak second order logic defines the natural numbers up to isomorphism, it does not have the upwards Skolem-Löwenheim Property. However, weak second order logic does have the downwards Skolem-Löwenheim Property, contrary to monadic and full second order logic.[97] Thus, a second order formula is f-valid iff it is true in all countable structures. One consequence of this is that full second order logic cannot be interpreted in weak second order logic.[98]

The downwards Skolem-Löwenheim Property of weak second order logic also implies that the f-validity of each second order formula φ is reducible to the truth in $\mathbb{N}$ of a second order formula φ' of the form $\forall X\,\varphi_0[X]$ (X ranging over *all* sets), where φ_0 is a first order number theoretic formula. The idea is that φ_0 arrises from φ by interpreting each relational variable as ranging over numeric codes of finite relations over X. Thus, the set of f-valid second order formulas is Π^1_1, in contrast to the set of valid second order formulas, which is not definable even in type theory (Theorem 5.6.3).

Although weak second order theories are natural and are related to applications of logic in the theory of computing and in artificial intelligence, not very much has been proved about them. Two examples: the weak second order theory of linear ordering is decidable [Laüchli, 1968], and the weak second order theory of one function is decidable [Rabin, 1969].

---

[95] Recall that $\varphi_{\mathrm{suc}} \equiv_{\mathrm{df}} \forall x\,\mathbf{s}(x) \neq \mathbf{0} \wedge \forall x,y\,(\,\mathbf{s}(x) = \mathbf{s}(y) \rightarrow x = y\,)$.

[96] For example, $\mathit{Plus}(x,y,z) \leftrightarrow \exists R.(\ R(x,y,z) \wedge \forall u(\,R(x,\mathbf{0},u) \leftrightarrow u \mathbin{=\!\!=} x\,) \wedge \forall u,v\,(\,R(x,\mathbf{s}u,\mathbf{s}v) \rightarrow R(x,u,v)\,)\,)$; $\mathit{Times}(x,y,z) \leftrightarrow \exists R.(\,R(x,y,z) \wedge \forall u(\,R(x,\mathbf{0},u) \leftrightarrow u \mathbin{=\!\!=} \mathbf{0}\,) \wedge \forall u,v\,(\,R(x,\mathbf{s}u,v) \rightarrow \exists w\,(\,R(x,u,w) \wedge \mathit{Plus}(x,w,v)\,)\,)$.

[97] A proof is sketched in [Monk, 1976] (pp. 489–490), where the result is attributed to Tarski, or see [Ebbinghaus *et al.*, 1984] exercise IX.2.7.

[98] Weak second order logic is trivially interpretable in standard second orer logic, because the notion of finiteness is second order definable (Section 2.2).

6.4 Predicative logic: Restricted comprehension

One of the main forces that have shaped the development of twentieth century mathematical logic was the crisis of foundations caused by the antinomies discovered around the turn of the century.[99] Several of the leading mathematicians of the time (such as Hilbert, Brouwer, and Poincaré) called for a careful reconstruction of Mathematics on safer grounds. One reconstructionist approach attempted to design formal calculi that would be powerful enough to capture as much mathematical practice as possible, while at the same time clearly avoiding the antinomies; this led to formalisms such as the ramified type theory of Whitehead and Russell and the formal set theory of Zermelo and Fraenkel.[100]

A more radical reaction to the crisis of foundations argued for a reconstruction of mathematics on foundations that are accepted, on the basis of a critical examination, as conceptually infallible, and not merely as avoiding the antinomies. One strain of the radical reconstructionist program was Brouwer's Intuitionism, rooted in a general critique of the notion of 'existence' of mathematical objects, and of the classical rules of logic that govern reasoning about existence.[101] However, as soon as intuitionistic logic was formalized by Heyting, it appeared that classical logic can be interpreted in it (see Section 4.3 above), implying that intuitionism does not play a central role in the foundational aspect of the reconstructionist program, notwithstanding its importance in other respects.

One is, therefore, led to focus on the other major strain of the reconstructionist program, the *predicativist program,* which proposes to examine critically not the logical rules governing reasoning about existence *in general,* but rather the existence of *particular* mathematical constructions, notably basic existence axioms: instances of the Comprehension Schema (Section 4.1), and instances of the axiom of choice, say in the numeric form of choice from numbers to sets:[102] $\forall x \exists Q\, \varphi[x, Q] \to \exists Z \forall x\, \varphi[x, Z_x]$. When referring to choice over formulas that mention set quantifiers, these set-existence principles are circular, since the existence of the set Z is based

[99] See e.g. the introduction of [Mendelson, 1964], or [Fraenkel *et al.*, 1973, Ch. 1], for concise and informative surveys of that crisis. [Fraenkel and Bar-Hillel, 1958, Section I.6] contains a detailed bibliography through 1956.

[100] [Whitehead and Russell, 1929; Zermelo, 1908; Zermelo, 1930; Fraenkel, 1922].

[101] The notion of 'existence' is understood here to encompasses disjunction, since, for example, $p \vee q$ is equivalent to $\exists x\, (x = 0 \to p) \wedge (x \neq 0 \to q)$.

[102] This form easily implies the schema of choice from numbers to numbers. Both these forms of choice follow from the general axiom stated in Section 4.2 by comprehension, but it is useful to consider particular instances of Choice when only weak forms of comprehension are present. Another numeric statement that follows from the set-theoretic axiom of choice is the schema of *dependent choice:* $\forall x \forall R \exists Q\, \varphi[x, R, Q] \to \exists Z\, Z_0 = \emptyset \wedge \forall x\, \varphi[x, Z_x, Z_{x+1}]$. Here, as in the schema of choice from numbers to sets, R and Q are unary, Z is binary, and Z_i is defined as in Section 4.2.

on the truth of φ, which presumes a meaningful quantification over the collection of all sets, including Z.

A definition of a collection C is said to be *impredicative* if it uses quantification whose range includes C as an element, i.e. if C is assumed to exist before it is defined.[103] Unrestricted impredicative definitions lead to contradictions, as in Russell's Paradox.[104] It is generally accepted that no contradiction occurs when a definition is used to carve out a subset of a given set, as formalized by the schema of Separation of Zermelo–Fraenkel Set Theory: $\exists x \forall y (y \in x \leftrightarrow \varphi \wedge y \in a)$, where x, a are not free in φ; that is, the set $\{y \in a \mid \varphi\}$ is legitimate. This restrictive use of Comprehension is justified by the conviction that every subset of a exists before it is ever defined. In particular, all definitions of real numbers are legitimate, even if they use quantification over all real numbers. An example of such a definition is the least upper bound of a set of reals.[105]

This ontology is, of course, anathema to the constructivist. The most basic form of circumventing impredicative definitions is to restrict Comprehension to first order formulas. This restriction is of interested, for one, due to the following:

Theorem 6.4.1. *Let T be a first order theory, and let T^2 be the theory in second order logic with comprehension over first order formulas, whose axioms are those of T. Then T^2 is conservative over T for first order formulas, i.e. every first order theorem of T^2 is a theorem of T.*

A simple proof of the theorem runs along the following lines. If φ is a theorem of T^2, then it has a cut-free sequential proof (see Section 3.3). It is easy to see that if Π is a cut-free proof in T^2 of a first-order formula, and Π' results from Π by deleting all second order formulas, then Π' is essentially[106] a proof of φ in T.[107] An extension of this will be discussed

---

[103] The term *impredicative* is due to Poincaré [1910].

[104] Suppose that C is the collection of all sets which fail to be elements of themselves; if C were a legitimate set, then it is defined in terms of quantification whose range include C, leading to the contradictory equivalence $C \in C \Leftrightarrow C \notin C$.

[105] Each real can be identified with the Dedekind cut that defines it. Then, the l.u.b. b of a bounded set A of reals is defined as $\{r \in \mathbb{Q} \mid (\exists a \in A)(r \in a)\}$, i.e. one uses Comprehension: $\exists b \, \forall r \in \mathbb{Q} (r \in b \leftrightarrow (\exists a \in A)(r \in a))$.

[106] Depending on the exact formulation of the sequential calculus, Π' may need to be slightly repaired to yield a correct proof

[107] [Fraenkel *et al.*, 1973] (p. 132 fn. 2) attributes this use of cut-free proofs to Paul Cohen. A proof theoretic proof that does not depend on cut elimination runs as follows. (See [Troelstra, 1973] for details, for the case of first order arithmetic and the schema of induction.) Suppose Δ is a proof in T^2 of a first order formula ψ. Then Δ can use only finitely many instances of comprehension, for formulas $\chi_1 \ldots \chi_m$ say. Δ can then be converted into a correct first order proof Δ', by interpreting each second order quantifier as ranging over the sets defined by substitution instances of $\chi_1 \ldots \chi_m$. Each formula in Δ is thus converted into a first order formula, the interpretation of the comprehension schema is provable in T, and the derived formula ψ is unchanged under the interpretation, concluding the proof. Yet another, model theoretic proof, which is

in Section 7.1 below (Theorem 7.1.1).

Theorem 6.4.2. *Let T^2 be as above. Then every theorem of T^2 can be proved using comprehension for universal first order formulas only.*[108]

Proof. We show, by induction on the formula φ, that each instance of comprehension for first order formulas, $\exists R \forall \vec{x}\,(R(\vec{x}) \leftrightarrow \varphi)$, is provable using comprehension for universal formulas only. It suffices to treat formulas in which the only logical constants are $\wedge$, $\neg$ and $\forall$. If φ is quantifier free then it is a special case of a universal formula, and the theorem holds trivially. If φ is of the form $\varphi_0 \wedge \varphi_1$ then, by induction assumption, there are relations R_0, R_1 that are co-extensional with φ_0 and φ_1, respectively. Applying the schema above to the formula $R_0(\vec{x}) \wedge R_1(\vec{x})$ for φ, we obtain the desired R. If φ is of the form $\neg\varphi_0$, the proof is similar. Finally, suppose φ is of the form $\forall y \varphi_0$. Then, by induction assumption, there is a relation $R_0(\vec{x}, y)$ coextensional with φ_0. Using comprehension for the universal formula $\forall y R_0(\vec{x}, y)$, we obtain the desired relation R. ∎

The use of comprehension can be further reduced to special cases, such as the following (which will be further refined in Section 7.2).[109]

Theorem 6.4.3. *Let V be a finite vocabulary, and let $\mathcal{F}_n[V]$ consist of the first order V-formulas all of whose textual subformulas have at most n free variables. Then there is a finite number of instances of comprehension from which all instances of comprehension for formulas in $\mathcal{F}_n[V]$ are derivable.*

Moreover, if pairing and projection functions are available in a theory T, then comprehension is reducible in T^2 to a finite number of instances.

Proof. Consider the following instances of comprehension which, as in the proof of Theorem 6.4.2, permit the derivation of comprehension for a compound formula from comprehension for its components:

1. For every $k \leq n$, $\forall S \exists R \forall x_1 \ldots x_k\ (\ R(\vec{x}) \leftrightarrow \neg S(\vec{x})\ )$
2. For every $k \leq n$, $\forall S_0, S_1 \exists R \forall x_1 \ldots x_k\ (\ R(\vec{x}) \leftrightarrow S_0(\vec{x}) \wedge S_1(\vec{x})\ )$
3. For every $k \leq n$, $\forall S \exists R \forall x_1 \ldots x_k\ (\ R(\vec{x}) \leftrightarrow \forall y S(y, \vec{x})\ )$

It remains to list finitely many instances of comprehension that imply (possibly using comprehension for compound formulas) every atomic instance of comprehension, namely:

---

due in the case of set theory to [Novak, 1951; Rosse and Wang, 1950; Mostowski, 1951], runs as follows. If φ is not a theorem of T, then (by the Completeness Theorem for first order logic) there is a model of $T \cup \{\neg\varphi\}$. This model can be extended to a Henkin model of T^2, and so φ cannot be a theorem of T^2. The model theoretic proof has the disadvantage of not providing an effective method for converting a proof in T^2 into a proof in T.

[108] This theorem, as well as its proof presented here, seem to belong to the folklore of the subject, at least for second order arithmetic.

[109] The idea of the proof seems to be due to von Neumann [1925]. See Section 7.2 for an application.

4. $\exists R \forall x, y \, (\, R(x,y) \leftrightarrow x{=}y \,)$,
5. $\exists R \forall \vec{x} \, (\, R(\vec{x}) \leftrightarrow \mathbf{Q}(\vec{x}) \,)$, for every relation constant $\mathbf{Q}$ of V;
6. $\exists F \forall \vec{x}, y \, (\, F(\vec{x},y) \leftrightarrow \mathbf{f}(\vec{x}){=}y \,)$, for every function constant $\mathbf{f}$ of V;
7. For every i, k, where $i < k \leq n$,
$$\forall Q^k \exists R^k \forall \vec{x} \, (\, R(x_1 \ldots x_i x_{i+1} \ldots x_k) \leftrightarrow Q(x_1 \ldots x_{i-1}, x_{i+1}, x_i, x_{i+2}, \ldots x_k) \,).$$
8. For every $k \leq n$, $\forall Q^k \exists R^k \forall x_1 \ldots x_k \, (\, R(x_2 \ldots x_k) \leftrightarrow Q(x_1 \ldots x_k) \,)$.

If an injective pairing function $\langle \cdot, \cdot \rangle$ is present, then coding of tuples of all arities becomes available, and it suffices to have the instances above for $k = 3$, provided we add an instance of comprehension that permits going back and forth between pairs and coded pairs:

$$\forall Q \exists R \forall x, y (R(\langle x, y \rangle) \leftrightarrow Q(x,y) \,)$$
$$\text{and} \quad \forall R \exists Q \forall x, y (R(\langle x, y \rangle) \leftrightarrow Q(x,y) \,)$$

∎

Since Comprehension restricted to first order formulas is conservative over first order logic (Theorem 6.4.1), this restriction is a rather extreme measure for avoiding impredicativity. However, the underlying idea of this restriction can be further iterated. We stipulate that relations fall into levels, with the base level consisting of those relations whose definition involves no relational quantification, i.e. first order definable relations. The next level consists of sets whose definition may use quantification over sets of the base level, and so on. This eliminates circularity, since in a set $S = \{n \mid \forall X \, \varphi\}$, if the set-variable X ranges over sets of level k, then that range excludes S, since $level(S) > k$.

The idea of stratifying abstraction into levels goes back to the Ramified Type Theory of [Russell, 1908; Whitehead and Russell, 1929], whose purpose was precisely to circumvent the antinomies of Naive Set Theory.[110] It is present in many mathematical, logical and philosophical development, notably in the predicative development of Analysis (see below) and of Set Theory[111]. Let us outline the resulting formalisms.

The simplest manifestation of the idea of stratifying properties can be found in a predicative variant of second order logic, *(finitely) ramified second order logic* [Church, 1956]. One posits variables ${}^{\ell}R^k$ for k-ary relations of level ℓ. The intended semantics is defined inductively, and is intertwined with the syntax of the language. Given a V-structure $\mathcal{S}$ with universe U, let ${}^{\ell+1}U^k$ consist of the relations $X \subseteq U^k$ definable by a formula in which all variables are of level $\leq \ell$, where ${}^{m}R^j$ $(m \leq \ell)$ are interpreted as ranging over ${}^{m}U^j$. We say that a formula φ of predicative second order logic is

[110] See [Hazen, 1983] for a survey of ramified type theories.

[111] See e.g. [Quine, 1937; Quine, 1951; Wang, 1954; Wang, 1962]

true in a structure $\mathcal{S}$ if it is true in $\mathcal{S}$ when variables ${}^{\ell}R^k$ are interpreted as ranging over ${}^{\ell}U^k$.

Defining a ramified form of finite order logic is slightly more delicate, and requires that we intertwine levels with the types themselves. The types are defined as follows: $(\iota, 0)$ is a type of level 0 (the type of individuals); if $\tau_1 \ldots \tau_n$ are types, of levels $\ell_1 \ldots \ell_n$ respectively ($n \geq 0$), and if $\ell > \ell_1, \ldots, \ell_n$, then $((\tau_1 \ldots \tau_n), \ell)$ is a type, of level ℓ.[112] The idea is that the collective reference to all objects of a certain level ℓ is of level $> \ell$, just as is the case for quantification over such a collection. For each leveled type τ we posit an unbounded supply of variables R^{τ}, intended to range over objects of type τ. The level of a formula φ is now defined to be the smallest number larger than all levels of bound variables in φ, and $\geq$ the levels of the free variables therein.

The construction of levels can be further extended into transfinite ordinals, by taking at limit ordinals ξ the union over lower levels: the leveled type (τ, ξ) is the union of (τ, ℓ) with $\ell < \xi$. However, the predicative nature of the ordinals used becomes an issue, and the purely logical nature of the language and its intended interpretation are increasingly in question. On the other hand, on conceptual grounds it is hard to defend stopping the ramified type hierarchy at any particular level. We comment further about this uneasy balance in Section 7.2 below.

An interesting issue is the ontological status and practical interest of ramified higher order logics. While some philosophers have argued for the importance of these formalisms, others have noted that, for all its technical machinery, ramified higher order logic is no stronger that plain first order logic in differentiating between structures:[113] if $\mathcal{S}$ and $\mathcal{Q}$ are two V-structures that satisfy the same first order sentences (where V is a given vocabulary), then they satisfy the same V-sentences of ramified finite order logic. Notwithstanding this similarity, the expressive power of even low levels of ramified second order logic is far greater than that of first order logic; consequently, the ramified second order theories of certain structures are more complex than the corresponding first order theories.

An important case in hand are systems for Predicative Analysis (Section 7.3), based on the ramified second order theory of the natural numbers. In particular, a truth definition for first order arithmetic can be obtained by using quantification over first order definable relations (i.e. the lowest level

[112] In particular, we have the propositional types $((), \ell)$ for all $\ell \geq 0$.

[113] Among the promoters of ramified higher order logic are Hacking and Hazen [Hacking, 1979; Hazen, 1983; Hazen, 1985]. The following observation is due to Sundholm [1981]: it follows from the fact that ramified higher order logic satisfies the conditions of Lindström's Theorem (Section 5.2), namely compactness and Downward Skolem-Löwenheim, by work of Leblanc [1976], and is therefore equivalent in the sense considered to first order logic.

in the ramified hierarchy of relations).[114] Another interesting example of the expressive power of ramified higher order logic over particular structures is the interpretability of Robinson's Arithmetic (an undecidable theory) in level 2 ramified second order theory of dense linear orders [Hazen, 1992]. This ought to be contrasted with the first order theory of dense linear orders, which is decidable.

7 Mathematical practice

Mathematics is replete with second order notions and images. These can often be contorted into first order molds, but at considerable costs, conceptual as well as technical. Jon Barwise referred to the gulf between first order logic and mathematical practice in these words: *'As logicians, we do our subject a disservice by convincing others that logic is first order, and then convincing them that almost none of the concepts of modern mathematics can really be captured in first order logic'* [Barwise, 1985]. A detailed compendium of even the more important higher order constructions in mathematical practice is well beyond the scope of this survey. Instead, we propose to consider the general issue of second order axioms versus first order schemas for principles such as Induction and Replacement. We then briefly comment on second order aspects of set theory and analysis, and finally we describe the speed up of proofs by use of higher order means. We shall not consider the many important uses of higher order constructs in various other fields, such as Geometry, Algebra, Topology, and Recursive Function Theory.

7.1 Second order axioms vs. first order schemas

Several central mathematical principles which are inherently second order are commonly approximated in first order logic by axiom schemas, i.e. by templates for an infinite collection of formulas. Two important examples

[114] This is analogous to the construction of truth definitions in Section 2.6. We can state that a set T contains a pair $(\#\varphi, x)$, where φ is in prenex-disjunctive normal form, only if φ is true under the valuation coded by x:

$$\begin{aligned}
\tau_0'[T] \quad \equiv \quad & \forall u,v,x\,(T(eq(u,v),x) \;\rightarrow\; val(u,x) = val(v,x)) \\
& \wedge\, \forall u,v,x\,(T(\neg eq(u,v),x) \;\rightarrow\; val(u,x) \neq val(v,x)) \\
& \wedge\, \forall u,v,x\,(T(disj(u,v),x) \;\rightarrow\; T(u,x) \vee T(v,x)) \\
& \wedge\, \forall u,v,x\,(T(conj(u,v),x) \;\rightarrow\; T(u,x) \wedge T(v,x)) \\
& \wedge\, \forall u,v,x\,(T(univ(u,v),x) \;\rightarrow\; \forall n\, T(u, inst(x,v,n))) \\
& \wedge\, \forall u,v,x\,(T(exst(u,v),x) \;\rightarrow\; \exists n\, T(u, inst(x,v,n)))
\end{aligned}$$

where *conj* and *exst* are additional primitive recursive functions that code the conjunction and existential quantification operations. Now, a prenex-disjunctive normal formula φ is true iff $\exists T(\tau_0'[T] \wedge \forall x T(\overline{\#\varphi}, x))$, where the existential quantifier ranges over arithmetic predicates.

are the Induction Schema in first order arithmetic, and the Replacement Schema in set theory.

The schema of induction, $\varphi[\mathbf{0}] \wedge (\,\forall u\, \varphi[u] \to \varphi[\mathrm{s}(u)]\,) \to \forall x \varphi$, is implied, using comprehension (for φ), by the second order formula $\forall x \forall R\ R(\mathbf{0}) \wedge (\,\forall u\ R(u) \to R(\mathrm{s}(u))\,) \to R(x)$, i.e. from $\forall x\, N[x]$.[115] In first order axiomatizations of arithmetic, such as Peano's Arithmetic (PA), the single second order axiom $\forall x\, N[x]$ is replaced by all (or some) of its first order specializations. Clearly, this is not an ideal rendition of $\forall x\, N[x]$, which is the intended statement. One believes in the correctness of all instances of induction because one believes in the second order axiom, not on the basis of an examination of its individual instances. Moreover, the rendition of induction by a schema is sensitive to variation in the language: as the language expands, so does the collection of instances of induction.[116]

Related observations apply to the principle of *transfinite induction* over well-founded relations. The well-foundedness of a binary relation R is expressed by

$$WF[R] \ \equiv_{\mathrm{df}}\ \forall X\,(\,(\exists z\, X(z)) \to (\exists z\,(\, X(z) \wedge \forall y\,(R(y,z) \to \neg X(y))\,)\,)$$

Taking the contrapositive of the scope of $\forall X$, and using comprehension for $X \equiv \neg Q$, we obtain transfinite induction over R:

$$TI[R] \ \equiv_{\mathrm{df}}\ \forall Q\ (\,\forall z\,(\,\forall y\,((R(y,z) \to Q(y)) \to Q(z)\,)\,) \to \forall z\, Q(z).$$

In first order arithmetic neither $WF[R]$ nor $TI[R]$ can be even expressed directly. Indeed, when one says that PA proves transfinite induction for a numeric relation R, one means that every specialization of $TI[R]$ with a first order formula,

$$TI[R,\varphi] \ \equiv_{\mathrm{df}}\ (\,\forall z\,(\,\forall y\,((R(y,z) \to \varphi[y]) \to \varphi[z]\,)\,) \to \forall z\, \varphi[z].$$

is provable.

Our second example of a schema is Replacement. The common Zermelo–Fraenkel axiomatization of Set Theory has a finite number of axioms, plus the Replacement Schema (due to Fraenkel): for each (first order) formula φ in the language,[117]

$$(\forall x \in y\, \exists! z\, \varphi[x,z]) \to \exists w (\forall x \in y)\,(\exists z \in w)\, \varphi[x,z],$$

that is, if φ defines the graph of a class F, whose restriction to a set y is a univalent mapping, then F maps y to a set w.

---

[115] $\varphi[t]$ abbreviates here $[t/x]\varphi$, the result of simultaneously substituting t for all free occurrences of x in φ.

[116] These and related issues are discussed in [Kreisel, 1967].

[117] We write $\varphi[t,s]$ for $[t,s/x,y]\varphi$, where x,y are some distinct variables

The practical interest in theories such as Peano's arithmetic and ZFC is their axiomatizability by a finite number of axioms and schemas. Thus, they are finitely axiomatizable as soon as second order quantification is allowed, namely:

Theorem 7.1.1. *Let T be a first order theory axiomatized by axioms $\alpha_1 \dots \alpha_k$, and by the first order instances of an axiom schema $\sigma[\varphi]$.*[118] *Let T^2 be the theory, in second order logic with comprehension for first order formulas, with the finitely many axioms $\alpha_1 \dots \alpha_k$, and $\forall R \sigma[R]$. Then T^2 is conservative over T.*[119]

The proof is straightforward, using Theorem 6.4.1. Special cases of this theorem are: second order arithmetic with the second order induction axiom and first order (respectively, recursive) comprehension is conservative over first order arithmetic (respectively, primitive recursive arithmetic). Similarly, ZF based on second order logic, with Replacement formulated as a second order axiom and with comprehension for first order formulas, is conservative over ZF.

This should be contrasted with the non-finite axiomatizability of the corresponding *first* order theories:

Lemma 7.1.2. [Kreisel, 1968; Kreisel and Levy, 1968] *Let T be a first order theory that proves cut-elimination for first order logic (via a suitable coding of the syntax), has for each k a provable truth definition for all formulas of complexity $\leq k$, and proves induction for all formulas in the language (in an interpreted form, at least). Then T is not finitely axiomatizable.*

Proof Sketch. Let $\alpha_1 \dots \alpha_k$ be formulas in the vocabulary in hand. Given a formula φ, let $n(\varphi)$ be the complexity of of $(\alpha_1 \wedge \dots \wedge \alpha_k) \rightarrow \varphi$. Since cut-elimination is provable (via coding) in T, T also proves that if $(\alpha_1 \wedge \dots \wedge \alpha_k) \rightarrow \varphi$ is provable, then it has a proof Π in which all formulas are of complexity $\leq n(\varphi)$. For such formulas T has a truth definition. Using induction with respect to the length of Π, for that truth definition, T then proves that all formulas φ deduced from $\alpha_1 \dots \alpha_k$ are true. Thus, if T were axiomatized by $\alpha_1 \dots \alpha_k$, then T would have proved reflection for T, contradicting Gödel's Incompleteness Theorem. ∎

It can be shown that both PA and ZF satisfy the conditions of the Lemma, and are therefore not finitely axiomatizable as first order the-

[118] Here φ is the syntactic parameter for the substituted formula.

[119] The theorem generalizes trivially to any number of schemas, and also to the case where schemas are restricted to a particular class $\mathcal{F}$ of first order formulas (the same for all schemas), such as the existential formulas, and with comprehension in T^2 also restricted to $\mathcal{F}$.

ories.[120] The proof above also shows that if one takes second order arithmetic with first order comprehension but with induction as a *schema*, for *all* formulas in the language, then the resulting theory is *not* conservative over first order arithmetic. Similarly, second order ZF with the induction schema (i.e. transfinite induction up to ω) for *all* second order formulas is not conservative over ZF.

7.2 Higher order aspects of set theory: from higher order to first order and back

The primacy of first order logic in foundational studies is well justified by central properties of first order logic such as completeness and the first order axiomatizability of most all classes of algebraic structures. However, these qualities are of limited relevance to the mathematics of canonical structures such as the natural numbers. On the one hand, there is no complete axiomatization even for the quantifier-free number-theoretic formulas true in the standard model. On the other hand, even the mathematics of natural numbers makes extensive uses of higher order objects, as in Analytic Number Theory.[121] One wonders, then, whether the widespread insistence on avoiding direct reference to higher order constructs is justified.

One source of that insistence is, paradoxically, the very acceptance of higher order constructs as a generic process that can be iterated. Whitehead and Russell, in their renowned *Principia Mathematicae* [Whitehead and Russell, 1929], developed much of the Mathematics of their time on the basis of a variant of finite order logic, *Ramified Type Theory*, in an attempt to reduce Mathematics to Logic.[122] One problem in formalizing Mathematics in Ramified Type Theory is that the underlying concepts may be viewed as begging further extension. By iterating the power-set construction to transfinite levels, one gets the cumulative universe of sets: Taking a basic universe V_0 of basic, 'uninterpreted', objects, let $V_{\alpha+1} =_{\rm df} \mathcal{P}(V_\alpha)$ for any ordinal α, and $V_\lambda =_{\rm df} \cup_{\alpha \prec \lambda}$, for limit ordinals λ. But this itera-

[120] The non-finite axiomatizability of first order arithmetic was proved in [Ryll-Nardzewski, 1953; Rabin, 1961]. An argument somewhat analogous to the above was used in [Levy, 1960] to show that the schema of Replacement cannot be axiomatized by a finite number of its instances. Indeed, if φ is a given formula in the language of ZF, then ZF proves, using a suitable instance of Replacement, that there is a large enough set X such that φ is true iff φ^X (i.e. φ relativized to X) is true, see [Levy, 1960]. It follows that ZF proves the consistency of any finitely axiomatizable subtheory of ZF. See e.g. [Cohen, 1966, Section II.8] for an exposition.

[121] From the discussion in Section 7.3 below, it follows that many of the methods of Analytic Number Theories can be re-coded within Peano's Arithmetic; however, not all methods are amenable to such coding, and, moreover, the coding is done at a considerable cost to expository naturalness.

[122] In the introduction to *Principia* they claim to have reduced mathematical axioms to logical principles, saying: *'What were formerly taken, tacitly or explicitly, as axioms, are either unnecessary or demonstrable'* ([Whitehead and Russell, 1929, p. v], quoted in [Andrews, 1986]).

tion is tantamount to viewing the power-set operation as a mathematical, rather than a logical operation; in particular, it cannot be captured in the syntax.[123] It is then natural and technically attractive to amalgamate all levels, leading to various first order theories for the universe $\cup_{\alpha \text{ an ordinal}} V_\alpha$ with the membership relation, of which ZF is the best known. Thus, the primacy of first order logic as a logical calculus and the adoption of set theory as a universal formalization of mathematics are, conceptually as well as historically, intertwined.

It is telling, therefore, that second order versions of ZF have re-emerged as being of interest both as frameworks for formalizing set theory and other parts of mathematics, and as relevant to the meta-theory of set theory itself.[124] One motivation for these extensions is the fundamental conflict between the importance of comprehension in our intuitive understanding of the concept of set [125] and the exclusion of comprehension from ZF, so as to avoid the set-theoretic paradoxes. The compromise achieved in ZF is to weaken comprehension to the principle of Separation,[126] according to which if a is a given set, and $\varphi[x]$ is a formula of the language of ZF, then there is a set $b = \{x \in a \mid \varphi\}$.

The main rationale of second order set theories is to permit at least reference to arbitrary collections, even when these are of 'unmanageable size'. That is, the intended first order objects of these theories are sets, and the second order sets (unary relations) then range over a broader collection of 'classes'. Every set a is also a class,[127] because by Comprehension $\exists A\, \forall x\, (A(x) \leftrightarrow x \in a)$. But some classes, such as $\{x \mid x = x\}$, are not sets, and are dubbed *proper classes*. Because both first order and second order objects are 'classes of sets,' it is customary to refer to second order extensions of ZF and to related theories as *theories of classes*, and to use the $x \in a$ notation ambiguously for both the first order membership relation between sets, and for the relational application $a(x)$ (when a is viewed as

[123] However, as observed by Kreisel and Feferman (see e.g. [Feferman, 1977]), the transfinite iteration of the type hierarchy has barely any connection with the actual development of mathematical analysis, which is in practice restricted to small, let alone finite, types. Among the few exceptions is the use of transfinite types to prove the determinacy of Borel games; see e.g. [Martin, 1975].

[124] A detailed survey of these issues can be found in Section II.7 of [Fraenkel *et al.*, 1973].

[125] A famous quote is the informal definition of a set by Cantor's the creator of abstract set theory: *A set is a collection into a whole of definite distinct objects of our intuition or of our thought* [Cantor, 1895].

[126] The Principle of Separation is often called the Subset Axiom. Zermelo's original phrase was *Axiom der Aussonderung,* literally: the axiom of singling-out. This weakening of Comprehension is a facet of the *doctrine of size,* according to which the paradoxes are due to the presence of sets of 'unmanageable size'. A more radical critique of Comprehension underlies the theory of semi-sets [Vopénka and Hájek, 1972].

[127] More precisely, a is co-extensional with some class.

a class).[128]

Among the theories of classes there are variants of second order ZF with comprehension restricted to first order formulas, and variants with full comprehension. To the first group belong the formalisms of von Neumann [1925], Bernays [1937; 1958], Gödel [1940] and Mostowski [1939]. In one style (e.g. [Gödel, 1940]) the second order character of the theory is masked: the theory is rephrased as a first order theory of classes, in which the notion of a set is definable (x is a set iff $x \in a$ for some class a), and comprehension is restricted to formulas where all quantifiers are bounded to sets. Let us denote by GBN the theories in this group. By Theorem 6.4.1 the theories GBN are conservative over ZF. By Theorems 7.1.1 and 6.4.3 GBN are finitely axiomatizable, not only as second order theories (capturing the schemas of subset and replacement by single axioms), but also as first order theories (capturing comprehension by a finite number of instances thereof).

Similar statements apply to analogous second order extensions of other first order set theories, such as Z (= ZF without replacement), and ZFC (= ZF plus the axiom of choice). Note that in GBN we may state the principle of *global choice*, asserting the existence of a class acting as a global choice function for all sets in the universe: $\exists C \forall x\, (\, x \neq \emptyset \rightarrow \exists! y \langle x, y\rangle \in C \wedge y \in x)$.[129]

The other main group of theories of classes, consisting of variants of second order ZF with *full* comprehension, includes the theories of Wang [1949], Quine [1951], Tarski [1981], Kelley [1955], and Morse [1965]. Let us denote by KM the theories in this group. The KM theories are particularly attractive for actual formalization of branches of mathematics with a strong set theoretic component, notably of point set topology [Kelley, 1955] and category theory. The relation of KM to ZF is analogous to the relation of full analysis to first-order arithmetic: KM proves reflection for ZF, and is therefore not a conservative extension thereof. An argument similar to the proof of Lemma 7.1.2 shows that KM is not finitely axiomatizable.[130]

Higher order logic has also impacted the development of set theory itself, notably concerning the status of strong infinity axioms. The most important results here were inspired by the Montague–Lévy reflection principle [Levy, 1960], which posits that if a formula φ is true in the universe of sets, then φ must be true when relativized to some universe in the cu-

[128] An exception is [Bernays, 1958], where the latter relation is denoted $x \eta a$.

[129] This axiom is conservative over [GBN + Choice] for first order formulas [Felgner, 1971], but is not provable in it [Easton, 1964].

[130] If KM′ is a theory axiomatized by a finite number of theorems of KM, then KM proves reflection for KM′, and so KM is not finitely axiomatizable. Reflection for KM is provable assuming the existence of a Mahlo cardinal [Levy, 1960], but the consistency of KM is reducible already to the consistency of ZF + there exists an inaccessible cardinal [Fraenkel *et al.*, 1973, p. 139].

mulative hierarchy, that is φ^{V_α} is true, for some cardinal α. The principle is interiorized as an axiom schema in a very powerful extension of KM, due to Bernays [1976].[131] Various forms of this principle imply the existence of ever larger cardinals, including measurable cardinals, which do not seem to have a clear justification on the basis of *first* order closure conditions.[132]

7.3 Analysis and reductive proof theory

The ubiquitous presence of infinite constructions in Analysis, and some early confusion about the inference rules governing them, were the main motivations for efforts in the late 19th century to formalize Analysis, leading to the adoption of second order arithmetic as a canonical formalism, e.g. in the influential monograph of Hilbert and Bernays [1939].[133] Although the formalization of mathematics and the discussion of formalizability were based mostly on set theory during the middle third of the century, some logicians, notably Kreisel and Feferman, have consistently argued for a more direct formalization, within second order arithmetic.[134]

Within logic the interest in second and higher order arithmetic has been stimulated, for one, by the emergence of recursion theory in higher types, which is closely related to higher order arithmetic in language, methods, and results.[135]

The study of higher order arithmetic was also stimulated by the continued interest in *predicative* mathematics, whose formulation is fundamentally related to second order arithmetic. The predicativist program aims at examining the development of Analysis with predicative forms of the principles above, or with forms that can be predicatively justified on the basis of a metamathematical analysis. The main issues addressed are: (1) Delineate formalisms whose predicative nature is brought out clearly, and

---

[131] See [Chuaqui, 1981] for a textbook development of set theory based on Bernays's axioms.

[132] Applications of this form are in [Gloede, 1976]. The existence of measurable cardinals is not provable in Bernays's theory [Solovay *et al.*, 1978], but it is provable in a further natural extension of the theory, allowing hyper-classes [Marshall, 1989]. Another example linking second order logic to higher cardinals is the Hanf number of second order logic. If L is a logic, and ψ a sentence of L, let $h(\psi)$ be the largest cardinality of a model of ψ, if such cardinality exist, and let $h(L)$, the *Hanf number of* L, be the supremum of all well-defined cardinals $h(\psi)$. Thus, every sentence ψ with a model of size $h(L)$ must have models of arbitrarily large size. Barwise [Barwise, 1972] showed that the Hanf number of second order logic cannot be proved to exist without using Replacement for a formula φ that uses universal quantification for all sets in the universe.

[133] Poincaré summarized this achievement at the 1900 Congress of Mathematicians: *'Today there remain in analysis only integers and finite or infinite systems* [=sets] *of integers ... Mathematics has been arithmetized ... We may say today that absolute rigor has been achieved'* ([Poincaré, 1902], quoted in [Fraenkel *et al.*, 1973, p. 14]).

[134] See e.g. [Feferman, 1977] for a broad and informative survey of the development of mathematical analysis in finite order logic.

[135] See e.g. [Kechris and Moschovakis, 1977] for a survey and further references.

classify their proof theoretic strength; (2) Reduce seemingly impredicative methods to predicative ones, or otherwise justify them by predicative means; (3) Identify those parts of Analysis which can be formalized in various predicative calculi.

Formalisms for predicative analysis can be defined along several organizing principles, notably: (1) Ramified analysis of levels going up to some fixed, predicatively justified ordinal; (2) Ramified analysis for levels up to *any* well-ordering justified predicatively within the theory itself;[136] (3) Analysis with weak comprehension, but extended with transfinite induction over predicatively justified well orderings;[137] (4) Weak set existence principles;[138] (5) Theories of iterated comprehension;[139] and (6) Theories of iterated inductive definitions.[140]

There are simple-to-prove, yet instructive classification results for the proof theoretic strength of these theories; examples are the observations that second order arithmetic with comprehension and induction for first order (arithmetical) formulas only is conservative over Peano's Arithmetic, and that Zermelo's set theory without the power set axiom is mutually interpretable with Peano's Arithmetic.[141]

---

[136] Here the 'predicative acceptability' of a well-ordering is intertwined with the definition of the predicative formalism itself. This is achieved in Feferman's system IR by alternating the justification of new well-orderings with their use for transfinite induction, leading to a closure ordinal Γ_0, identified independently by Schütte. See [Feferman, 1964; Feferman, 1968; Schütte, 1965b; Schütte, 1965a].

[137] If $\prec$ is a binary relation on natural numbers, without an infinite descending chain (i.e. if $\prec$ is order-isomorphic to a countable ordinal), then *transfinite induction on* $\prec$ is the schema $(\forall x \in \text{field of } \prec)((\forall y \prec x \varphi[y]) \to \varphi[x]) \to (\forall x \in \text{field of } \varphi[x])$. See [Schütte, 1951; Schütte, 1952; Schwichtenberg, 1977] for these theories.

[138] The predicative status of set existence principles is not clear-cut, but particular weak principles are usually viewed as predicatively obvious, notably comprehension for arithmetic first order formulas (that is, the first level of the ramified hierarchy), and Weak König's Lemma, which states that a binary tree with arbitrarily long branches has an infinite branch. It is easily seen that Arithmetic comprehension is equivalent to the axiom of transfinite induction, often dubbed *Bar Induction*: $\text{WF}[X] \to \forall Y\ \text{TI}[X, Y]$, where $\text{WF}[X]$ states that X is a well founded binary relation, and $\text{TI}[X, Y]$ is transfinite induction over X for the unary predicate Y. (See, e.g., [Feferman, 1977, Section 6.1.5] for a proof.)

[139] For example, the theory ATR_0 of *arithmetical transfinite recursion* permits the definition of functions and sets by recursion of an arithmetic predicate, over a well-ordering. See e.g. [Friedman *et al.*, 1982; Simpson, 1982a; Simpson, 1982b] for detailed expositions. Note: (1) the principle is a schema, with arithmetic instances; (2) the principle is an implication, of the form *If* the binary relation R is a well-ordering, *then* transfinite recurrence over R with respect to the arithmetic formula $\mathbf{f}$ is allowed.

[140] See [Buchholz *et al.*, 1981]. Such definitions are a paradigm of predicative, bottom-up, construction of subsets of $\mathbb{N}$. Intuitionistic theories of transfinitely iterated inductive definitions were introduced in [Kreisel, 1963]. A detailed account of such theories is in [Buchholz *et al.*, 1981].

[141] Two predicative formalisms may have the same proof-theoretic power, as measured by provable well-orderings, yet differ substantially in their ability to capture mathematical practice.

The second among the three predicativist projects listed above is the predicative justification of seemingly impredicative methods. An early and famous example of a failed attempt of this kind is Hilbert's Program, which called for securing mathematics by showing, using 'finitistic methods' only, that the 'finitistic' theorems of Analysis are true.[142] Gödel's Incompleteness Theorems imply that this goal is far too ambitious, because finitistic mathematics cannot prove reflection even for provability of $0 = 1$ in finitistic mathematics itself. Nonetheless, extensive proof theoretic work has resulted in reductions of seemingly impredicative methods of Analysis to more constructive ones.[143]

The third predicativist project is the explicit development of analysis within predicative theories, and the calibration of the predicativity/constructivity level of particular mathematical results. An early example is the development of much of analysis using only arithmetic comprehension, by Weyl [1918].[144] Sieg notes that virtually all of the turn-of-the-century analysis, as presented in [Hilbert and Bernays, 1939], can be formalized already in second order arithmetic with Π^1_1 comprehension and induction for Π^1_1 formulas [Sieg, 1990]. However, as one would expect from the reductionist program outlined above, much of analysis can be carried out in formalisms weak enough to be conservative over Peano's Arithmetic (see e.g. [Friedman, 1977; Friedman, 1980]). To calibrate the 'degree of predicativity' of particular theorems one shows that a theorem is not only a consequence of a principle of predicative analysis, but is in fact equivalent to such a principle. This project, initiated by H. Friedman and pursued by

[142] In more technical terms: (1) formalize 'finitistic mathematics', i.e. the part of mathematics which deals with concrete objects only, in particular the natural numbers. (A related adage is the famous saying of Kronecker's, that *God has created the integers, everything else is man's imagination.*) This is usually taken to be primitive recursive arithmetic. (2) Prove within finitistic mathematics reflection for provability of finitistic (i.e. quantifier free) formulas within second order arithmetic.

[143] Important examples of which are: (1) A consistency proof of full classical analysis using higher order recursive functionals defined by Bar Induction [Spector, 1962]. (2) With respect to Π^1_3 formulas, choice for Σ^1_2 formulas is conservative over comprehension for Δ^1_2, and the latter is conservative over the theory of iterated inductive Π^1_1-definitions of length up to ϵ_0 [Friedman, 1970; Buchholz *et al.*, 1981]. (3) With respect to Π^1_2 formulas, choice for Σ^1_1 formulas is conservative over comprehension for Δ^1_1, and the latter is conservative over ramified analysis of levels up to ϵ_0 [Friedman, 1967]. Moreover, Δ^1_1 sets are definable by a natural transfinite extension of the arithmetical hierarchy [Kleene, 1955], providing further constructive justification for them. (4) Weak König's Lemma is conservative, with respect to Π^1_1 formulas, over second order arithmetic with recursive comprehension and Σ^0_1-induction [Sieg, 1987]. Weaker or alternative versions of the latter result have been proved since [Friedman, 1969]. In particular, a proof theoretic reduction of Weak König's Lemma to primitive recursive arithmetic, of a general nature, can be found in [Sieg, 1985]. Some surveys related to these are contained in [Kreisel, 1965; Kreisel, 1968; Kreisel, 1971; Feferman, 1977; Buchholz *et al.*, 1981; Simpson, 1985b; Sieg, 1990].

[144] An exposition of more modern parts of analysis along the same lines is [Zahn, 1978].

S. Simpson and his students, has been known as *reverse mathematics.*[145] It turns out that great many theorems of analysis are equivalent[146] to one of four principles, which in increasing proof theoretic strength are: (1) Weak König's Lemma, (2) comprehension for arithmetic formulas, (3) arithmetic transfinite recursion (ATR, see above), and (4) comprehension for Π^1_1 formulas.

A project dual to delineating the predicative nature of theorems of analysis consists in identifying theorems about the natural numbers (and therefore of a concrete, 'finitistic', nature) which are *essentially impredicative*, i.e. which have no predicative proof.[147]

7.4 Speed-up

7.4.1 Speed-up of formalisms

Our discussions above suggest that the use of higher order logic for number and set theories is epistemologically more honest, and methodologically cleaner, than the more restrictive use of first order schemas. A natural question is whether such extensions of the logic result in gain in length of proofs. This notion of *proof speed-up* is quite general, but it depends on the yardstick used to measure proofs.

First, the definition refers to proof calculi, and not to theories (set of theorems): if $\mathcal{F}'_0$ is a formalism that has the theorems of $\mathcal{F}_0$ as axioms, then every theorem of $\mathcal{F}_0$ has a one line proof in $\mathcal{F}'_0$, and meaningful speed-up over $\mathcal{F}'_0$ is impossible. It is therefore important to restrict attention to natural formalisms, with a suitably simple set of axiom.[148] Also, we can

[145] The equivalence of 'theorems' to 'axioms' is, as noted in [Sieg, 1990], an old theme, however. Already Dedekind showed that major theorems of analysis are equivalent to the topological completeness of the reals [Dedekind, 1872]; also, a well-known compendium of theorems of mathematics that are equivalent to the Axiom of Choice is [Rubin and Rubin, 1966]. See [Simpson, 1986] for a survey of the Reverse Mathematics project. The seminal papers of the project are [Friedman, 1975; Friedman, 1976; Steel, 1976], and other key papers are [Friedman *et al.*, 1982; Friedman *et al.*, 1983; Simpson, 1984; Simpson, 1986; Brown and Simpson, 1986].

[146] The equivalences are proved within a weak base theory, in which comprehension and induction are allowed over recursive predicates only.

[147] By Gödel's Incompleteness Theorem, the consistency of second order arithmetic is a Π^0_1 sentence which cannot be proved in second order arithmetic, let alone using predicative methods only. However, such consistency sentences are arithmetically coded metamathematical statements, and their combinatorial (concrete) nature is not evident. Better examples are the unprovability in ATR_0 of Kruskal's Theorem about tree embeddings, and the unprovability of an extended form of Kruskal's Theorem in analysis with comprehension and induction for Π^1_1-formulas. A detailed survey of these results is [Simpson, 1985a]. The seminal work on combinatorial statements independent of Peano's arithmetic is [Paris and Harrington, 1977]. Kruskal's Theorem asserts that in every infinite sequence of finite trees there must occur a tree within which some previous tree in the list can be embedded. Friedman [1981] showed that neither this theorem, nor a Π^0_2 'finite form' thereof, are provable in ATR_0. The results about an extended form of Kruskal's Theorem are in [Friedman, 1982].

[148] One usually requires that the set of axioms be finite, or generated by a finite number

measure a proof (for the definition of speed-up) either by counting the number of steps (lines) in it,[149] or the number of symbols. We refer to these as the proof's *length* and its *size,* respectively. An essential difference between proofs' length and size is that there can be only finitely many proofs with a given size, but infinitely many of a given length. If φ is derivable in $\mathcal{F}$ then we denote by $pflth_{\mathcal{F}}(\varphi)$ (respectively, $pfsz_{\mathcal{F}}(\varphi)$) the smallest n such that φ has a proof in $\mathcal{F}$ of length n (respectively, size n). If all the theorems of a formalism $\mathcal{F}_0$ are among the theorems of a formalism $\mathcal{F}_1$, and h is a unary numeric function, then we say that $\mathcal{F}_1$ *has a proof-length speed-up over* $\mathcal{F}_0$ *by a factor of* h if there is an infinite set Φ of theorems of $\mathcal{F}_0$, such that for every $\varphi \in \Phi$, $pflth_{\mathcal{F}_0}(\varphi) > h(pflth_{\mathcal{F}_1}(\varphi))$. The definition of proof-size speed up is analogous. If there is an infinite set Φ of theorems of $\mathcal{F}_0$ such that, for some fixed n, $pflth_{\mathcal{F}_1}(\varphi) < n$ for all $\varphi \in \Phi$, but the values $pflth_{\mathcal{F}_0}(\varphi)$ with $\varphi \in \Phi$ exceed any bound, then we say that $\mathcal{F}_1$ *has an unbounded proof-length speed-up over* $\mathcal{F}_0$.[150]

The speed-up resulting from using increasingly higher order constructs can be observed in two related yet somewhat different realms: theories for arithmetic (more generally, theories for a particular structure, in which the syntactic machinery can be coded), and pure logics. We review below results for both.

7.4.2 Proof-size speed-up of higher order arithmetic

Our first and simplest setting is proof-size speed-up of k+1 order arithmetic over k order arithmetic, with respect to first order theorems. Let PA_k be k order Peano Arithmetic (PA), that is the k order theory axiomatized by Peano's axioms for $\mathbf{0}$, $\mathbf{s}$, $+$ and $\times$, and with induction and comprehension for all formulas in the language. Our choice of logical rules matters little: any one of the customary Hilbert-style, natural deduction, or sequential calculi will do.

Theorem 7.4.1. *Let Z, Z' be sound extensions of PA, where Z' proves reflection for Z. Then, for every computable (total) function h, Z' has a proof-size speedup over Z by a factor of h. In particular, PA_{k+1} has a proof-size speedup over PA_k by a factor of h.*

Proof. Let $H(n) =_{\mathrm{df}} \sum_{i \le n} h(2 \cdot i)$. Then H is a total recursive function, and for every c, $H(n) > h(c+n)$ for all $n \ge c$. By Gödel diagonalization method[151] there is a Π^0_1 formula $\varphi[x]$, with one free variable x, such that

---

of schemas.

[149] That is, the number of formulas, or annotated formulas, or sequents, depending on the formalism in hand.

[150] An analogous definition for proof size is of no interest: since there are only finitely many formulas with proofs of size $\le n$, there is no infinite Φ such that $pfsz_{\mathcal{F}_1}(\varphi) < n$ for all $\varphi \in \Phi$.

[151] See e.g. [Smorynski, 1977, Section 2.2.1].

the following equivalence (suitably coded) is provable in PA.

$$\varphi[n] \quad \leftrightarrow \quad \forall y \ (\text{ if } H(n) \text{ computes to } y \text{ then } \chi[y,n] \ )$$

where

$$\chi[y,n] \quad \equiv_{\mathrm{df}} \quad \varphi[\bar{n}] \text{ has no proof in } Z \text{ of size } < y.$$

The formula $\varphi[\bar{n}]$ is true for each n, for if $\varphi[\bar{n}]$ is false, then $H(n)$ has some output y, and there is a Z-proof of $\varphi[\bar{n}]$ of size $< y$. But all theorems of Z are true, so $\varphi[\bar{n}]$ is true, a contradiction.

Since reflection for Z is provable in Z', the argument above is formalizable in Z', i.e. $Z' \vdash \forall x \varphi$. Let $\psi_n =_{\mathrm{df}} \varphi[\bar{n}]$. This is derivable in a single step from $\forall x \varphi$, so ψ_n has a proof in Z' of size $c_0 + n$, where c_0 is a constant uniform for all n, namely the size of the proof of $\forall x \ \varphi[x]$.

Each ψ_n is also a theorem of Z, in fact of PA: since H is total, there is a calculation of $H(n) = m$ for some m, which can be verified in PA. Also, in PA it can be proved that $H(n)$ can have at most one output. So, reasoning in Z, if $H(n) = z$, then $z = m$. The fact that $\varphi[\bar{n}]$ has no proof of size $< m$ is also a calculation verifiable in PA, so PA proves the r.h.s. of the equivalence defining φ. Since that equivalence is proved in PA, we conclude that $\varphi[\bar{n}]$ follows within PA. This proves $\underline{\psi_n}$.

Finally, since ψ_n is true, so is the formula $\chi[\overline{H(n)}, \bar{n}]$, i.e. ψ_n has no proof in Z of size $< H(n)$.

We have proved that ψ_n has a proof of size $c_0 + n$ in Z', and is provable in Z but not by any proof of size $< H(n)$, whence by no proof of size $< h(c_0 + n)$, for all $n > c_0$. This proves the theorem. ∎

Proofs of Theorem 7.4.1 can be found in [Mostowski, 1952] and [Ehrenfeucht and Mycielski, 1971]. In [Buss, 1992] a proof is given in which Φ consists of the independently interesting formulas $Con_k(\bar{n})$, where $Con_k(\bar{n})$ is (a coded form of) the statement *'PA_k has no proof of $0 = \bar{1}$ of size $\leq n$.'* In contrast to Theorem 7.4.1, the size speed-up of second order arithmetic with arithmetic comprehension over first order arithmetic is polynomial.[152]

Note that the speed-up above is for universal formulas. For existential theorems, speedup of $k+1$ order arithmetic over k order arithmetic is somewhat more tame:

Theorem 7.4.2. *Let Z, Z' be sound extensions of PA, where Z' proves reflection for Z. For every provably recursive function h of Z', Z' has a proof-size speed-up over Z of a factor of h, for Σ^0_1 theorems.*[153]

[152] This follows from the proof of [Troelstra, 1973] for the arithmetic case of Theorem 6.4.1, mentioned in footnote there.

[153] **Proof sketch:** Let g be a strictly increasing provably recursive function of Z' such that, for every s, if Δ is a proof in Z of an existential sentence $\exists y \psi[y]$, of size $\leq s$, then there is some $m < g(s)$ such that $\psi[\bar{m}]$ is true. Such g exists, because Z' proves reflection

However, Theorem 7.4.2 does not extend to speed-up by arbitrary recursive functions:

Theorem 7.4.3. *Let Z be a sound extension of PA, and let Z' prove reflection for Z. If Z has a proof-size speed-up over PA of factor h, for Σ^0_1 sentences, then h is majorized by a provably recursive function of Z'.*

Proof. Let g be the function that, on input s, returns the longest quantifier free proof of any Σ^0_1 sentence with a proof in Z of size $\leq s$. Then g is provably total in Z', and g majorizes h. ∎

7.4.3 Proof-length speed-up of higher order arithmetic

The earliest statement about proof speed-up seems to have been announced (without proof) by Gödel [1932]: for every recursive function h, PA_{k+1} has a proof-length speed-up over PA_k by a factor of h. This has been substantially improved recently by Buss [1992]:[154]

Theorem 7.4.4. *For each k, PA_{k+1} has an unbounded speed-up over PA_k. Moreover, the speed-up is for Π^0_1 formulas.*

The formulas for which speed-up is obtained are generated as follows. Let $\varphi[x]$ be a formula such that

$$\varphi[x] \leftrightarrow \text{`}\varphi[\bar{x}] \text{ has no proof in } PA_k \text{ of length} \leq x\text{'}$$

is provable in PA. Then one considers the formulas $\varphi[\bar{n}]$, for $n \geq 0$.

7.4.4 Speed-up of higher order logic

The speed-up results above for number theories do not translate to analogous results for logical formalisms, because those speed-ups refer to formulas whose combinatorial complexity (in PA) exceeds the expressive capability of pure first order logic. Indeed, higher order logics have a more

---

for Z, witnesses for true Σ^0_1 sentences can be extracted effectively, and the number of proofs proofs of size n as a function of n is provable in PA. If h is provably recursive in Z', then $Z' \vdash \forall x \exists y, z\ T(\bar{e}_h, 2x, z) \wedge T(\bar{e}_g, U(z), y)$. (Here T is Kleene's computation predicate, U is Kleene's primitive recursive result-extracting function, and e_g and e_h are codes for algorithms for g and h, respectively.) Say the proof for the above has size s_0. Then the formula $\varphi_n \equiv_{\text{df}} \exists v\ T(\bar{e}_h, 2 \cdot \bar{n}, (v)_0) \wedge T(\bar{e}_g, U((v)_0), (v)_1)$ has a proof in Z' of size $s'_0 = n + s_0 + c$, for some constant c. (If T is represented by a formula of PA, then the coefficient of n in s_0 is > 1.) Note that if v is the (unique) value for which the matrix above is true, then $(v)_1 = g(h(2n))$. But if Δ is a derivation in Z of φ_n, of size s_1, then, by definition of g, $v < g(s_1)$, and so $(v)_1 = g(h(2n)) < g(s_1)$. Since g is strictly increasing, this implies, for all sufficiently large n, and assuming without loss of generality that h is non-decreasing, that $s_1 > h(2n) > h(n + s_0 + c) \geq h(s'_0)$.

[154] Buss's theorem applies to formalizations of higher order arithmetic that are 'weakly schematic,' i.e. where each axiom is either a tautology, a universal closures of a tautology, or an instance of one of finitely many schemas. Parikh [Parikh, 1973; Parikh, 1986] gave a proof of Theorem 7.4.4 for speed-up of PA_2 over PA_1, which was extended to speed-up of PA_{k+1} over PA_k by Krajíček [Krajíček, 1989]. However, both proofs refer to formalizations where addition and multiplication are ternary relations.

tame speed-up over first order logic than do higher order number theories over first order arithmetic. Let $\mathcal{L}_i$ be the natural deduction formalism for i order logic, as described in Section 4.1.[155] First, observe that there is no proof-size speed-up by factors that are too fast-growing:

Theorem 7.4.5. *There is a provably recursive function h of third order arithmetic such that $pfsz_{\mathcal{L}_1}(\varphi) \leq h(\, pfsz_{\mathcal{L}_2}(\varphi)\,)$.*

Proof Sketch. Third order arithmetic proves that every theorem of second order logic is valid in all structures. Since the Completeness Theorem for first order logic is provable in second order arithmetic, it follows that third order arithmetic proves that every first order theorem of second order logic is provable in $\mathcal{L}_1$. Thus, the following algorithm for h is provable in third order arithmetic: for input s, enumerate all $\mathcal{L}_2$ proofs of size $\leq s$, search for a first order proof of each theorem encountered, and take the maximum over the size of these proofs. ∎

Consequently, for h as above it is not the case that $\mathcal{L}_2$ has a proof-size speed-up over $\mathcal{L}_1$ by a factor of h.

An analogous result for proof-length is due to Statman:[156]

Theorem 7.4.6. *There is a provably recursive function h of third order arithmetic such that, for all first order formulas φ, $pflth_{\mathcal{L}_1}(\varphi) \leq h(\, pflth_{\mathcal{L}_2}(\varphi)\,)$. Consequently, $\mathcal{L}_2$ does not have a proof-length speed up over $\mathcal{L}_1$ of factor h.*

In contrast to the situation with higher order number theories, using higher order logic rather than second order logic has negligible effect on proof-length speed-up for first order theorems. The proof is an easy application of Theorem 5.6.1 and its proof:

Theorem 7.4.7. *For each $k \geq 3$ there are linear functions h_s, h_ℓ such that, for every first order φ,*

$$pfsz_{\mathcal{L}_2}(\varphi) \leq h_s(\, pfsz_{\mathcal{L}_\omega}(\varphi)) \qquad \text{and} \qquad pflth_{\mathcal{L}_2}(\varphi) \leq h_\ell(\, pflth_{\mathcal{L}_\omega}(\varphi))$$

For *second* order theorems the converse is true: since first order arithmetic formulas are expressible as second order formulas, it follows from Theorems 7.4.1 and 7.4.4 that, for second order formulas, $\mathcal{L}_{k+1}$ has proof-size speed-up over $\mathcal{L}_k$ of any recursive factor, as well as unbounded proof-length speedup.

---

[155] Let $\mathcal{L}_i^=$ be the extension of $\mathcal{L}_i$ with the equality rules for atomic formulas. All the speed-up results stated below for the logics $\mathcal{L}_i$ hold equally for the logics $\mathcal{L}_i^=$.

[156] Theorem 7.4.6 is proved in [Statman, 1978, Sections 6.2 and 6.4]. The proof there yields a provably recursive function of third order arithmetic, because the proof itself is formalizable in third order arithmetic.

While Theorems 7.4.5 and 7.4.6 indicate the limits of possible speed-up by higher-order logic, second order logic still has a rather formidable speed-up over first order logic:

Theorem 7.4.8. *For every provably recursive function h of second order arithmetic, $\mathcal{L}_2$ has a proof-size speed-up over $\mathcal{L}_1$ by a factor of h.*

The proof is similar to the proof of Theorem 7.4.2.[157]

8 Higher order logic in relation to computing and programming

We propose to briefly comment on a few of higher order logic's fundamental relations to computing and programming: the very use of higher order data, the computational nature of natural deduction for higher order logic, the use of higher order logic in the meta-theory of formal systems, and the relations between second and higher order logic and computational complexity. This selection is intended to give an idea of the kinds of interactions between higher order logic and computer science, and not to suggest that other topics are of lesser importance or interest. Most notable is our omission of a legion of uses of higher order consructs in computer science, whose relation to logic is less direct.

8.1 Higher order data and types

Higher order functions, in the form of procedures, are at the core of higher level programming language, whether imprerative or applicative. Whereas in assembly languages the computation data (input and output) is restricted to finite stored information, modern high level languages have assigned from their inception an increasing role to programming constructs that manipulate *conceptual* entities, such as modules, procedures, and types. A landmark advance was the procedure definition facility of Algol 60 and its descendents, such as ADA, which use the entire type structure of finite order logic [Reynolds, 1972; Reynolds, 1981], as captured for example by the simply typed lambda calculus.[158] An explicit use of infinitely many types within the syntax of the language seems to have appeared

[157] The absence of induction in $\mathcal{L}_2$ is inconsequential, because second order arithmetic is interpretable in $\mathcal{L}_2$. One still considers $\varphi_n \equiv \psi[\bar{n}]$ where $\psi[x] \equiv \exists v\ T(\bar{e}_h, 2x, (v)_0) \wedge T(\bar{e}_g, U((v)_0))$. However, the proof of φ_n in $\mathcal{L}_2$ proceeds via proving the second order formula $\forall x\,(N[x] \to \psi[x])$.

[158] In functional languages, such as Lisp, Scheme, and ML, the lambda notation is integrated into the syntax of the programming language, and the type hierarchy is either implicit in a good programming style (Lisp and Scheme), or explicit in the language (ML). More recently, extensions of logic programming with higher order abstraction have also been developed and implemented. See e.g. [Mycroft and O'Keef, 1984; Nadathur, 1987; Miller and Nadathur, 1986; Miller, 1988; Miller, 1989; Lakshman and Reddy, 1991; Yardeni *et al.*, 1991], and in particular [Nadathur and Miller, 1990]. These topics are covered in [Miller, 1993] in this Handbook.

first in Algol 68.[159] More complex programming concepts related to the type hierarchy (yet not to abstraction for types) include inheritance (and the related constructs of coercion and subtyping) and type intersection; these have been developed recently, notably in relation to object oriented programming languages.[160]

Once types are explicitly integrated into the syntax of programming languages, it becomes clear that applying various forms of abstraction to the types themselves is both natural and useful. Types are then used both as notations for functional behaviour of data objects, enforcing a compile-time type checking, and as data. The interplay between these two roles can become rather involved, both computationally and semantically. In one guise of abstraction, types can depend on objects, as in 'the type of numeric arrays of length k'. This kind of abstraction underlies the type discpline of Martin-Löf [1973b], and of the AUTOMATH family of languages [Bruijn, 1980].

Another form of abstraction over types is polymorphism, which underlies (in a weak form) the programming language ML. Just as the functional type hierarchy is captured mathematically by the simply typed lambda calculus, polymorphism is captured by the *second order* lambda calculus, invented independently by Girard and Reynolds [Girard, 1972; Reynolds, 1974].[161] In the interest of our discussion in 8.2 below, let us outline the essentials of the second order lambda calculus. In the simply typed λ-calculus, one cannot have a single term representing constructions as basic as the identity function: for each type τ we have a distinct identity function[162] $I_\tau = \lambda x^\tau . x$. If we parametrize terms with respect to types then, using a variable t ranging over types, the term $\lambda x^t . x$ is a template for all the identity functions. However, t is not abstracted here within the syntax itself, so $\lambda x^t . x$ is not a term with a definite meaning, which can be passed as an argument to other terms. To achieve that, one introduces a *type abstraction* construct Λ, so that the *polymorphic* identity function can finally be defined as $I = \Lambda t . \lambda x^t . x$. The collection of types is enriched accordingly with a construct $\forall$, and $\Lambda t . \lambda x^t . x$ is declared to be of type $\forall t . t \to t$. A term whose type is of the form $\forall t . \sigma$ can be applied to a *type*

[159] John Reynolds, in personal communication, writes: *Whatever its faults, Algol 68 represents the first realization by a computer scientist that a typed higher-order programming language must have an infinite number of types, that this type structure is part of the syntax of the language, and that it must be described schematically.*

[160] For instance, the type intersection discipline of [Coppo and Dezani-Ciancaglini, 1980] has been incorporated into Reynold's programming language Forsythe [Reynolds, 1988], and inheritance into Cardelli's language Quest [Cardelli, 1991; Cardelli and Longo, 1991].

[161] See e.g. [Scedrov, 1990] for a survey. The model theory of second order lambda calculus is a fascinating albeit complex story. See e.g. [Scedrov, 1990] and [Gunter, 1992] chapter 11 for general expositions, and references there for pointers to the technical literature.

[162] We use an optional type supserscript to indicate the type associated with a variable.

τ, to yield a term of type $[\tau/t]\sigma$.[163] For example, the term $(\Lambda t.\lambda x^t.\, x)\,\tau$ is of type $\tau \to \tau$. One also uses a form of β-reduction for types: an expresison $(\Lambda t.\, E)\tau$ reduces to $[\tau/t]E$; for example $(\Lambda t.\lambda x^t.\, x)\,\tau$ reduces to $I_\tau = \lambda x^\tau.\, x$. Thus, I is a polymorphic identity function, from which the specific identities I_τ can be recovered by type application and β-reduction for types.

The two forms of abstraction for types, with respect to objects and with respect to types, are merged in the Calculus of Construction [Coquand, 1985; Coquand and Huet, 1985; Coquand and Huet, 1988]. A further construct involving type abstraction is the recursion operator on types, which permits the introduction of user-defined types. This is a definable construct in the Theory of Construction, just as the fixpoint operator is definable in second order logic. The interested reader will find a thorough survey of types in programming languages in [Mitchell, 1990], and of typed lambda calculi in [Barendregt, 1992].

The development of higher order data has not been confined to programming languages. In database theory there has been a growing realization that many database applications have to do not only with functions and relations over simple objects, which can be describe in first order languages or mild extensions thereof, but also with complex hierarchical objects, whose formalization is natural in finite order logic. A familiar example is the hierarchical structure of files and directories under the UNIX operating system. Such objects are referred to as *complex*, and the resulting databases are *complex object databases.*[164] These database calculi refer to definition of queries for such objects using algebraic or logical operations.[165] An algebraic development of complex object databases uses basic operations on sets. One assumes a collection $\mathcal{Q}$ of *primitive sets*, and defines new sets using operations such as intersection, power set, union of elements, and projection. An alternative development uses simply sets defined explicitly by formulas of (a suitable variant of) finite order logic over the vocabulary $\mathcal{R}$ (extended by additional primitives if desired). The two approaches, suitably developed, turn out to be equivalent.[166]

---

[163] Here τ must be free for t in σ, in the usual sense.

[164] The idea of programming with higher order relations is not new. The rich theory of recursion in higher types aside, the use of higher order objects in database theory is probably due to Makinouchi [1977], and is related to the uses of higher order relations in the general purpose programming language SETL [Schwarts *et al.*, 1986].

[165] The *types* (also dubbed *sorts*) of a complex object database are taken to be the simple higher order relational types, built up using cartesian products with attributes as labels. That is, one assumes given a collection $\mathcal{D}$ of *basic types,* and a collection $\mathcal{A}$ of *attributes.* Then each $D \in \mathcal{D}$ is a type; if τ is a type then so is (τ); and if $\tau_1 \ldots \tau_k$ are types, and $A_1 \ldots A_k$ are distinct attributes, then $[\tau_1 : A_1, \ldots \tau_k : A_k]$ is a type. For each type τ, the collection U_τ of objects of type τ is defined by induction on τ: U_D, for $D \in \mathcal{D}$, is assumed given; $U_{(\tau)}$ consists of the finite subsets of U_τ; and $U_{[\tau_1:A_1,\ldots\tau_k:A_k]}$ consits of the k-tuples of pairs, $[u_1 : A_1 \ldots u_k : A_k]$, where $u_i \in U_{\tau_i}$.

[166] [Abiteboul and Beeri, 1988; Kuper and Vardi, 1984]. This result is essentially an

8.2 The computational nature of higher order natural deduction

In launching combinatory logic, Schönfinkel discovered that the basic inferences of logic are in fact of a simple functional nature [Schönfinkel, 1924]. This analogy was elaborated by Curry [Curry and Feys, 1958], and rediscovered by Howard in a yet more prestine form, for natural deduction proofs [Howard, 1980]. This duality reveals the fundamental unity between the operational aspects of logic and functional programming, and it is particularly fecund when extended to higher order forms of abstractions, namely dependent types (Martin-Löf's Type Theory [Martin-Löf, 1973b; Martin-Löf, 1980]) and finite order quantification (Girard's system F_ω [Girard, 1972]). Notably, its implementation underlies sofware systems for constructing typed functional programs for given specifications from higher order logic proofs that such specifications has solutions. These systems include the PRL Project [Constable and others, 1986], which is based on [Martin-Löf, 1973b], and the Calculus of Constructions mentioned above [Paulin-Mohring, 1989b; Paulin-Mohring, 1989a].

Given its importance, let us describe the simplest non-trivial case of this duality, namely between minimal second order propositional logic and the Girard–Reynolds second order lambda calculus described above.[167] We consider minimal second order propositional logic $\mathbf{ML}_2^0$ (compare Section 4.3), with implication and universal quantification over propositions as the only logical constants. A natural-deduction calculus for $\mathbf{ML}_2^0$ has the following inference rules:

$$\rightarrow\mathbf{I}: \quad \frac{\begin{array}{c}[\chi^i] \\ \dots \\ \varphi\end{array}}{\chi \rightarrow \varphi} \quad (\text{occurrences } \chi^i \text{ of } \chi \text{ are closed}) \qquad \rightarrow\mathbf{E}: \quad \frac{\chi \rightarrow \varphi \quad \chi}{\varphi}$$

$$\forall^2\mathbf{I}: \quad \frac{\varphi}{\forall R.\varphi} \quad (R \text{ not free in assumptions}) \qquad \forall^2\mathbf{E}: \quad \frac{\forall R\, \varphi}{[\chi/R]\varphi}$$

We define a mapping κ from derivations in this calculus to terms of the second order lambda calculus, as follows. Let us use the same identifiers

extension to finite order logic of the algebraic development of first order logic, due to Tarski [Henkin *et al.*, 1971]. A survey of these and related advances can be found in [Abiteboul and Kanellakis, 1990]. A seminal paper on computability for higher order objects is [Dahlhaus and Makowsky, 1987].

[167] For more details and applications of this mapping see, for example, [Girard *et al.*, 1989] and [Leivant, 1990b].

for relational variables of second order propositional logic and for type variables of the second order lambda calculus. This convention results in a complete syntactic identity between formulas of the logic and types of the lambda calculus. The following κ maps a derivation Π of a formula φ from open assumption $\psi_1 \ldots \psi_k$, to a λ-expression $\kappa\Pi$, of type φ, and with free variables of types $\psi_1 \ldots \psi_k$.

$\Pi \equiv \quad \varphi^i$ (open assumption φ labelled by i)	$\kappa\Pi =_{\mathrm{df}} x_i^\varphi$ (variable of type φ)
$\Pi \quad \equiv \quad \dfrac{\begin{array}{c}[\chi^i] \\ \Delta \\ \varphi\end{array}}{\chi \to \varphi}$	$\kappa\Pi =_{\mathrm{df}} \lambda x_i^\chi.\kappa\Delta$
$\Pi \quad \equiv \quad \dfrac{\begin{array}{cc}\Delta & \Theta \\ \psi \to \varphi & \psi\end{array}}{\varphi}$	$\kappa\Pi =_{\mathrm{df}} (\kappa\Delta)(\kappa\Theta)$
$\Pi \quad \equiv \quad \dfrac{\begin{array}{c}\Delta \\ \varphi\end{array}}{\forall R.\varphi}$	$\kappa\Pi =_{\mathrm{df}} \Lambda R.\kappa\Delta$
$\Pi \quad \equiv \quad \dfrac{\begin{array}{c}\Delta \\ \forall R.\varphi\end{array}}{[\chi/R]\varphi}$	$\kappa\Pi =_{\mathrm{df}} (\kappa\Delta)\chi$

Clearly, if Π derives φ from $\psi_1^{i_1} \ldots \psi_k^{i_k}$, then $\kappa\Pi$ is of type φ, with free variables $x_{i_1}^{\psi_1} \ldots x_{i_k}^{\psi_k}$ (of types $\psi_1 \ldots \psi_k$, respectively).

The mapping κ not only reveals a reading of derivations as λ-terms, but shows that the computational behavior of derivations, under the canonical detour-elimination transformations (Section 4.4 above), is identical to the computational behavior of the corresponding λ-terms under β-reductions (for objects and for types):

$$\text{the reduction of } \dfrac{\begin{array}{cc}[\psi^i] & \\ \Delta & \\ \varphi & \Theta \\ \psi \to \varphi & \psi\end{array}}{\varphi} \quad \text{to} \quad \begin{array}{c}\Theta \\ [\psi^i] \\ \Delta \\ \varphi\end{array}$$

is mapped to the β-reduction of $(\lambda x_i^{\psi}.\, \kappa\Delta)(\kappa\Theta)$ to $[(\kappa\Theta)/x_i^{\psi}](\kappa\Delta)$, and

$$\text{the reduction of } \dfrac{\begin{array}{c}\Delta \\ \varphi \\ \forall R.\, \varphi\end{array}}{[\chi/R]\varphi} \quad \text{to} \quad [\chi/R]\Delta$$

is mapped to the β-reduction for types of $(\Lambda R.\, \kappa\Delta)\chi$ to $[\chi/R](\kappa\Delta)$.

8.3 Higher order logic in the meta-theory of formal systems

Proponents of the first order formalization of mathematics have had to address the plain fact that many mathematical constructions and deductions are higher order. One common response has been that full formalization of mathematics is in any event a mere idealization, never carried out in practice in full detail. This argument has been shattered by the very presence of computers. First, it has become possible to store, organize, verify, and retrieve fully formalized mathematical texts.[168] Moreover, it became possible not only to store and manipulate completely formalized proofs, but also to automatically generate many proofs.

Initially, research in automated theorem proving enforced the primacy of first order logic, for which methods such as resolution and paramodulation are complete. However, the work of Andrews and his students has shown that automated theorem proving can be naturally extended to higher order logic [Andrews *et al.*, 1984; Miller, 1983].[169] This extension is linked to generalization, from first order to higher order, of the major syntactic theorems of first order logic: Cut elimination and normalization [Takahashi, 1967; Girard, 1972; Prawitz, 1972], resolution [Andrews, 1971], Skolemization [Miller, 1987], unification [Huet, 1975], and Herbrand's Theorem [Miller, 1987].[170] Given the natural description of mathematics within higher order logic, higher order theorem proving has led, quite recently, to

[168] The ground-breaking work was the AUTOMATH family of languages [Bruijn, 1980], designed as a medium for a computerized compendium of mathematics within a higher order logic.

[169] See also e.g. [Gould, 1976; Jensen and Pietrzykowski, 1976; Petersson, 1982].

[170] An early presage of higher order theorem proving is [Robinson, 1969].

rapid advances in the development of user-friendly and interactive software environments, based in higher order logic, for the fully formalized development of mathematics [Constable and others, 1986; Felty and Miller, 1988; Felty, 1993; Dowek, 1991; Dowek *et al.*, 1991].

A challenging extension of user-friendly automated manipulation of the mathematical vernacular is the manipulation of programming itself, a field often dubbed *meta-programming*. Here even the (denotational) semantics of basic constructs, such as iteration and recursion, is based on higher order constructs.[171] Recent formalisms, referred to as *logical frameworks,* were proposed to represent formally both the syntax and the semantics of programs, as well as of formal rules for manipulating and verifying programs. These use higher order constructs extensively.[172]

8.4 Higher order logic and computational complexity

Machine-independent characterizations of computational complexity classes lend credence to their importance, provide insight into their nature, relate them to issues relevant to programming methodology and to program verification, suggest new tools for separating complexity classes, and offer concepts and methods for generalizing computational complexity to computing over arbitrary structures and to higher type functionals. Two types of such characterizations relate computational complexity to abstraction level in higher order: higher order database queries (i.e. global methods of finite model theory), and function provability in higher order logics.

A *query* is a 'global relation' over a collection of structures; that is, if V is a vocabulary, a (k-ary) V-**query** is a map that, to each V-structure $\mathcal{S}$ assigns a (k-ary) relation over the universe $|\mathcal{S}|$ of $\mathcal{S}$.[173] For example, if V_G consists of binary relation-constant $\boldsymbol{\rho}$, then the *transitive-closure* is the query that assigns to each V_G-structure $\mathcal{G}$ the transitive closure of $\boldsymbol{\rho}^{\mathcal{G}}$. Every V-formula φ (together with a list $\vec{v} = v_1 \dots v_k$ of variables that

[171] Denotational semantics has grown, of course, into a broad and central field in theoretical computer science. The seminal work was [Scott and Strachey, 1971]. Recent expositions include the survey [Mosses, 1990] and the textbooks [Tennent, 1991; Gunter, 1992].

[172] Some recent key papers are [Felty and Miller, 1988; Harper *et al.*, 1987; Huet and Lang, 1978; Hannan and Miller, 1988b; Hannan and Miller, 1988a; Howe, 1988; Harper *et al.*, 1989; Huet, 1986; Knoblock and Constable, 1986; Lee and Pleban, 1987; Paulson, 1987; Pfenning, 1989; Pfenning and Rohwedder, 1991; Weber, 1991].

[173] The notion that a formula determines a process that uniformly delineates subsets of structures is implicit already in early formalizations of Set Theory, for instance in Frege's Comprehension Principle and, in particular, in Fraenkel and Skolem's Axiom of Replacement. In relation to collections of first order structures the notion was used by Tarski's [Tarski, 1952] (Definition 1) and in [Barwise and Moschovakis, 1978]. The phrase *data base queries* is from [Chandra and Harel, 1980]. Other terms for it include *global relation* [Gurevich, 1987]. *generalized relations*, [Rougemont, 1987] (referring to [Barwise and Moschovakis, 1978]), *global predicates*, [Blass and Gurevich, 1986], *uniformly defined relations*, [Rougemont, 1987], *predicate*, [Leivant, 1987], and *predicate over oracles* [Cai and Furst, 1987].

include all the variables free in φ) defines the query $\lambda\vec{v}.\varphi$, that assigns to a V-structure $\mathcal{S}$ the extension of φ in $\mathcal{S}$, i.e. the collection of $\vec{a} \in |\mathcal{S}|^k$ such that $\mathcal{S}, [\vec{a}/\vec{v}] \models \varphi$. Similarly, a computation device over V-structures, such as an uninterpreted program scheme over V, defines a query when it is used as an acceptor. We can thus compare the expressive power of descriptive and computational means: a class Φ of formulas characterizes a computational complexity class $\mathcal{C}$ if the queries computable over *finite* structures by algorithms in $\mathcal{C}$ are exactly the ones definable by formulas in Φ.

In this framework one is drawn immediately to higher order formulas, because even simple queries, such as the transitive closure, are not first order definable [Aho and Ullman, 1979].[174] Using higher order constructs indeed yields a surprising array of descriptive characterizations of computational complexity, in particular for structures that are given with an order.[175] Most of the results here fall into two organizing principles: syntactic classes of higher order formulas, and variants of fixpoint constructs.

In a seminal result, Fagin [1974] and Jones and Selman [1974] proved that a query over ordered finite structures is defined by a program running in nondeterministic polynomial time (NP) iff it is defined by a purely existential second order formula.[176] From this it immediately follows that a problem is definable by a second order formula iff it is in the polynomial time hierarchy. Characterizations of complexity classes below NP, namely deterministic polynomial time (P) and non-deterministic log-space (NL), can be obtained in terms of universal second order formulas with natural restrictions on the matrix. Dually, classes broader than NP are characterized in terms of higher order formulas, which yield characterizations of poly-space and exponential time in third order logic. At the limit, the queries definable in full finite order logic are precisely the ones computable within Kalmar-elementary resources, i.e. using time and space which is k-fold exponential in the size of the input, for some k [Leivant, 1987].

The seminal result on characterization of complexity classes in terms of fixpoint queries states that a query over ordered finite structures is in P iff it is definable in the logic $FO\mu$ (first order logic with a monotone fixpoint operation, see Section 6.2) [Immerman, 1986; Vardi, 1982]. A natural fixpoint construct over finite structure is the *non-inflationary fixpoint* of [Abiteboul

[174] We showed in Section 2.4 the somewhat easier theorem that transitive closure is not first order definable when considering *all* structures, not only the finite ones.

[175] So that their preparatory coding as Turing machine input does not by itself enable the machine to recognize properties of the structure, such as the parity of its size, which could not be recognized descriptively. The role played by order on finite structures has been greatly clarified recently, see e.g. [Abiteboul *et al.*, 1992].

[176] The theorem stated here is Fagin's. Jones and Selman proved a similar and closely related result, that first order spectra are precisely the sets computable in NTime(2^{cn}), where the input is given in binary. The proof methods of [Fagin, 1974] and [Jones and Selman, 1974] are similar.

and Vianu, 1989]: if Φ is an operator on k-ary relations, then $\bar{\mu}R.\Phi(R)$ is $\Phi^m(\emptyset)$, where m is the smallest number such that $\Phi^m(\emptyset) = \Phi^{m+1}(\emptyset)$. The queries definable in first order logic extended with $\bar{\mu}$ are exactly the ones computable in polynomial space [Abiteboul and Vianu, 1989]. Other, nondeterministic, variants of fixpoints yield a characterization of the queries computable in exponential time, as well as additional characterization of poly-space and of NP [Abiteboul *et al.*, 1992].[177]

Another method of logical characterization for computational complexity classes uses proof theory, that is, the deductive machinery of logic rather than the descriptive machinery outlined above. Proof theoretic characterization are of particular conceptual interest, because proof principles codify directly conceptual abstraction, and are therefore the most natural medium for exploring and articulating a foundational justification for linking complexity classes with specific forms of conceptual abstraction. Proof theoretic characterizations of classes of computable functions follow the following pattern. Given a formalism F whose language provides a natural rendition of statements of the form *'program P converges for all input,'* one associates with F the class $C(F)$ of computable functions for which there exists a program that F proves to converge for all input. Second order logic is of special interest here, because comprehension is a natural yardstick for calibrating degrees of abstraction;[178] and because, by Theorem 2.5.1, function convergence can be stated in second order logics without presupposing any functions or predicates as given.

If L is a formalism for second order logic, we say, using the notations and basic facts in Sections 2.3,2.5, that a function f over a free algebra $\mathbb{A}$ is **provable in** L iff it is computed by some coherent equational program $(P, \mathbf{f})$ such that

$$\tilde{\forall} P \vdash_L A(\vec{x}) \rightarrow A(\mathbf{f}(\vec{x})).$$

For a class Φ of formulas, let $L_2(\Phi)$ be second order logic with comprehension for formulas in Φ.

Using this definition, one obtains a spectrum of characterization results for complexity classes, which calibrate computational complexity classes in terms of abstraction levels, as measured by comprehension. The interpretation of second order arithmetic in second order logic (Section 2.3) implies that the provable functions of L_2(all second order formulas) are precisely the provably-recursive functions of second order arithmetic.[179]

---

[177] Characterizations that fall into neither of the spectra above include characterizations of nondeterministic log-space [Immerman, 1987; Blass and Gurevich, 1986], of poly-time [Immerman, 1987], and of exponential time [Christen, 1974; Immerman, 1987].

[178] Recall the Reverse Mathematics program, described in Section 7.3.

[179] A simple method for dealing with Peano's third and fourth axioms is given in [Leivant, 1990b].

Using comprehension for computational (i.e. strict-Π^1_1, see Section 3.2) formulas, one obtains exactly the functions defined by recurrence in all finite type [Leivant, 1990a; Leivant, 1991b].[180] Comprehension for first order formulas yields exactly the Kalmar-elementary functions.[181]

The characterization by comprehension of complexity classes of practical interest is also possible. Indeed, the functions over $\mathbb{W}$ computable in deterministic polynomial time are exactly the functions provable using comprehension for positive existential formulas.[182] The significance of these characterizations is discussed in [Leivant, 1991a; Leivant, 1994].

Acknowledgments

I am greatly indebted for detailed and conscientious feedback on early drafts of the chapter from Sam Buss, Kees Doets, Jean-Yves Marion, Dale Miller, Yiannis Moschovakis, Alan Mycroft, Frank Pfenning, John Reynolds, and Stewart Shapiro. My sincere gratitude also goes to Jon Barwise, André Chuaqui, David Harel, Marcin Mostowski, Larry Moss, Steve Simpson, Wilfried Sieg, Gaisi Takeuti, and Jeff Zucker, for useful comments and advice. The shortcomings in the chapter, surely a legion, are of course entirely my responsibility.

References

[Abiteboul and Beeri, 1988] S. Abiteboul and C. Beeri. On the power of languages for manipulating complex objects. Research Report 846, INRIA, 1988.

[Abiteboul and Kanellakis, 1990] S. Abiteboul and P. Kanellakis. Database theory column: Query languages for complex object databases. *SIGACT News*, 21/2:9–18, 1990.

[Abiteboul and Vianu, 1989] S. Abiteboul and V. Vianu. Fixpoint extensions of first-order logic and datalog-like languages. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science*, pages 71–79, Washington, D.C., 1989. IEEE Computer Society Press.

[180] I.e., the provably recursive functions of Peano Arithmetic.

[181] More precisely, given any free algebra $\mathbb{A}$, the function over $\mathbb{A}$ computable in Kalmar-elementary resources are exactly the ones provable in second order logic with first order comprehension. This should be contrasted with the first order definable queries, a class strictly contained in NL [Immerman, 1987].

[182] Comprehension for positive quantifier free formulas yields the same class. A formula is **positive** if it contains no negation or implication. The notion of provability used here is a slight variant of the one defined above: when comprehension is so drastically limited, different second order renditions of $\mathbb{W}$ make a difference, because their equivalence can no longer be proved.

[Abiteboul *et al.*, 1992] S. Abiteboul, M. Vardi, and V. Vianu. Fixpoint logics, relational machines, and computational complexity. In *Proceedings of the Seventh IEEE Conference on Structure in Computational Complexity*, 1992.

[Aczel and Mendler, 1989] P. Aczel and N. Mendler. A final coalgebra theorem. In D. H. Pitt, D. E. Rydeheard, P. Dybner, A. M. Pitts, and A. Poigné, editors, *Proceedings of the 1989 conference on Category Theory and Computer Science*, Volume 389 of LNCS, pages 357–365. Springer-Verlag, Berlin, 1989.

[Aczel, 1977] P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 739–782. North-Holland, Amsterdam, 1977.

[Aho and Ullman, 1979] A. V. Aho and J. D. Ullman. Universality of data retrieval languages. In *Sixth Symposium on Principles of Programming Languages*, pages 110–117, 1979.

[Ajtai and Gurevich, 1987] M. Ajtai and Y. Gurevich. Monotone versus positive. *Journal of the ACM*, 34:1004–1015, 1987.

[Ajtai, 1979] M. Ajtai. Isomorphism and second order equivalence. *Annals of Mathematical Logic*, 16:181–203, 1979.

[Amiot, 1990] G. Amiot. The undecidability of the second order predicate unification problem. *Archive for Mathematical Logic*, 30:193–199, 1990.

[Andrews *et al.*, 1984] P. B. Andrews, D. Miller, E. Cohen, and F. Pfenning. Automating higher-order logic. *Contemporary Mathematics*, 29:169–192, 1984.

[Andrews, 1965] P. B. Andrews. *A Transfinite Type Theory with Type Variables.* North-Holland, Amsterdam, 1965.

[Andrews, 1971] P. B. Andrews. Resolution in type theory. *Journal of Symbolic Logic*, 36:414–432, 1971.

[Andrews, 1972a] P. B. Andrews. General models and extensionality. *Journal of Symbolic Logic*, 15:395–397, 1972.

[Andrews, 1972b] P. B. Andrews. General models, descriptions, and choice in type theory. *Journal of Symbolic Logic*, 37:385–394, 1972.

[Andrews, 1986] P. B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth through Proof.* Academic Press, New York, 1986.

[Barendregt, 1992] H. Barendregt. Lambda calculi with types. In S. Ambramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science,* Volume 2. Oxford University Press, 1992.

[Barwise and Moschovakis, 1978] J. Barwise and Y. Moschovakis. Global inductive definability. *Journal of Symbolic Logic*, 43:521–534, 1978.

[Barwise, 1969] J. Barwise. Applications of strict π^1_1 predicates to infinitary logic. *Journal of Symbolic Logic*, 34:409–423, 1969.

[Barwise, 1972] J. Barwise. The Hanf number of second order logic. *Journal of Symbolic Logic*, 37:588–594, 1972.

[Barwise, 1975] J. Barwise. *Admissible Sets and Structures.* Springer-Verlag, Berlin and New York, 1975.

[Barwise, 1977] J. Barwise, editor. *Handbook of Mathematical Logic.* North-Holland, Amsterdam, 1977.

[Barwise, 1979] J. Barwise. On branching quantifiers in English. *J. Phil. Logic*, 8:47–80, 1979.

[Barwise, 1985] J. Barwise. Model-theoretic logics: Background and aims. In J. Barwise and S. Feferman, editors, *Model Theoretic Logics.* Springer-Verlag, New York, 1985.

[Behmann, 1922] H. Behmann. Beiträge zur Algebra der Logic und zum Entscheidungsproblem. *Math. Ann.*, 86, 1922.

[Benthem and Doets, 1983] J. van Benthem and K. Doets. Higher-order logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic,* Volume I, pages 275–329. Reidel, Dordrecht, 1983.

[Bernays, 1937] P. Bernays. A system of axiomatic set theory. *Journal of Symbolic Logic*, 2:65–77, 1937.

[Bernays, 1958] P. Bernays. *Axiomatic Set Theory.* North-Holland, Amsterdam, 1958.

[Bernays, 1976] P. Bernays. On the problem of schemata of infinity in axiomatic set theory. In G.H. Müller, editor, *Sets and Classes*, pages 121–172. North-Holland, Amsterdam, 1976.

[Blass and Gurevich, 1986] A. Blass and Y. Gurevich. Henkin quantifiers and complete problems. *Theoretical Computer Science*, 32:1–16, 1986.

[Blass and Gurevich, 1987] A. Blass and Y. Gurevich. Existential fixed-point logic. In E. Börger, editor, *Logic and Complexity*, Volume 270 of LNCS, pages 20–36. Springer-Verlag, Berlin, 1987.

[Blass *et al.*, 1985] A. Blass, Y. Gurevich, and D. Kozen. A zero-one law for logic with a least fixpoint operator. *Information and Control*, 67:70–90, 1985.

[Boolos, 1975] G. Boolos. On second order logic. *J. of Philosophy*, 72:509–527, 1975.

[Brown and Simpson, 1986] D. Brown and S. Simpson. Which set existence theorems are needed to prove the Hahn-Banach theorem for separable Banach spacesΓ *Annals of Pure and Applied Logic*, 31:123–144, 1986.

[Bruijn, 1980] N. G. de Bruijn. A survey of the project AUTOMATH. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 579–606. Academic Press, London, 1980.

[Buchholz *et al.*, 1981] W. Buchholz, S. Feferman, W. Pohlers, and W. Sieg. *Iterated Inductive Definitions and Subsystems of Analysis:*

Recent Proof-Theoreitc Studies. Volume 897 of LNM. Springer-Verlag, Berlin, 1981.

[Buss, 1992] S. R. Buss. On Gödel's theorems on lengths of proofs I: Number of lines and speedup for arithmetics. Preprint, 1992.

[Cai and Furst, 1987] J.-Y. Cai and M. L. Furst. Pspace survives three-bit bottlenecks. In *Proceedings of the Second Annual Conference on Structure in Complexity*, pages 94–102, Los Angeles, 1987. IEEE Computer Society Press.

[Cantor, 1895] G. Cantor. Beiträge zur Begründung der transfiniten Mengenlehr I. *Math. Annalen*, 46:355–358, 1895.

[Cardelli and Longo, 1991] L. Cardelli and P. Longo. A semantic basis for Quest. *Journal of Functional Programming*, 1:417–458, 1991.

[Cardelli, 1991] L. Cardelli. Typeful programming. In E. J. Neuhod and M. Park, editors, *Formal Description of Programming Languages*, pages 431–507. Springer-Verlag, Berlin, 1991.

[Chandra and Harel, 1980] A. K. Chandra and D. Harel. Computable queries for relational data bases. *Journal of Computer and System Sciences*, 21:156–178, 1980.

[Chandra and Harel, 1982] A. Chandra and D. Harel. Structure and complexity of relational queries. *Journal of Computer and System Sciences*, 25:99–128, 1982. Preliminary version in *Twenty First Symposium on Foundations of Computer Science* (1980) 333–347.

[Chang and Keisler, 1973] C. C. Chang and J. Keisler. *Model Theory.* North-Holland, Amsterdam, 1973.

[Christen, 1974] C. A. Christen. Spektren und Klassen elementarer Funktionen. Ph.D. Thesis, ETH Zurich, 1974.

[Chuaqui, 1981] R. B. Chuaqui. *Axiomatic Set Theory: Impredicative Theories of Classes.* North-Holland, Amsterdam, 1981.

[Church, 1949] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1949.

[Church, 1956] A. Church. *Introduction to Mathematical Logic.* Princeton University Press, Princeton, 1956.

[Cohen, 1966] P. J. Cohen. *Set Theory and the Continuum Hypothesis.* Benjamin, New York, 1966.

[Compton, 1988] K. J. Compton. 0-1 laws in logic and combinatorics. In I. Rival, editor, *NATO Advanced Study Inst. on Algorithms and Order*, pages 353–383. Reidel, Dordrecht, 1988.

[Constable and others, 1986] R. Constable et al. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[Coppo and Dezani-Ciancaglini, 1980] M. Coppo and M. Dezani-Ciancaglini. An extension of basic functionality theory for λ-calculus. *Notre-*

Dame Journal of Formal Logic, 21:685–693, 1980.

[Coquand and Huet, 1985] T. Coquand and G. Huet. Constructions: A higher order proof system for mechanizing mathematics. In *EUROCAL85*, Volume 203 of LNCS, Berlin, 1985. Springer-Verlag.

[Coquand and Huet, 1988] T. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 76:95–120, 1988.

[Coquand, 1985] T. Coquand. Une Théorie des Constructions. Ph.D. Thesis, Université Paris VII, January 1985.

[Curry and Feys, 1958] H. B. Curry and R. Feys. *Combinatory Logic.* North-Holland, Amsterdam, 1958.

[Dahlhaus and Makowsky, 1987] E. Dahlhaus and J. A. Makowsky. Computable directory queries. In *Logic and Computer Science: New Trends and Applications (Rend. Sem. Mat. Univ. Pol. Torino)*, pages 165–197, Turin, 1987.

[Davis, 1965] M. Davis, editor. *The Undecidable.* Raven Press, New York, 1965.

[Dedekind, 1872] R. Dedekind. *Stetigkeit und irrationale Zahlen.* Braunschweig, Vieweg, 1872. English translation: Richard Dedekind, *Essays on the Theory of Numbers, Continuity and Irrational Numbers.* Open Court, Chicago, 1901.

[Dowek *et al.*, 1991] G. Dowek, A. Felty, H. Herbelin, G. Huet, C. Paulin-Mohring, , and B. Werner. The Coq proof assistant user's guide. Technical Report 134, INRIA, 1991.

[Dowek, 1991] G. Dowek. Démonstration automatique dans le Calcul des Constructions. Ph.D. Thesis, Université Paris VII, 1991.

[Dragalin, 1979] A. G. Dragalin. New kinds of realizability. In *Sixth International Congress for Logic, Methodology, and Philosophy of Science*, 1979. (Abstract).

[Easton, 1964] W. B. Easton. Powers of regular cardinals. Ph.D. Thesis, Princeton, 1964. Revised version in *Annals of Math. Logic* 1:139–178, 1970.

[Ebbinghaus *et al.*, 1984] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic.* Springer-Verlag, Berlin, 1984.

[Ehrenfeucht and Mycielski, 1971] A. Ehrenfeucht and J. Mycielski. Abbreviating proofs by adding new axioms. *Bull. Amer. Math. Soc.*, 77:366–367, 1971.

[Enderton, 1970] H. B. Enderton. Finite partially ordered quantifiers. *Zetisch. für Math. Logik u. Grundlagen der Mathematik*, 16:393–397, 1970.

[Fagin, 1974] R. Fagin. Generalized first order spectra and polynomial time recognizable sets. In R. Karp, editor, *Complexity of Computation*, pages 43–73. SIAM-AMS, 1974.

[Fagin, 1976] R. Fagin. Probability on finite models. *Journal of Symbolic Logic*, 41:50–58, 1976.

[Feferman, 1964] S. Feferman. Systems of predicative analysis. *Journal of Symbolic Logic*, 29:1–30, 1964.

[Feferman, 1968] S. Feferman. Systems of predicative analysis II. *Journal of Symbolic Logic*, 33:193–220, 1968.

[Feferman, 1977] S. Feferman. Theories of finite type related to mathematical practice, 1977. In [Barwise, 1977], pages 913–971.

[Felgner, 1971] U. Felgner. Comparison of the axioms of local and universal choice. *Fundamenta Math.*, 71:43–62, 1971.

[Felty and Miller, 1988] A. Felty and D. A. Miller. Specifying theorem provers in a higher-order logic programming language. In E. Lusk and R. Overbeek, editors, *Ninth International Conference on Automated Deduction*, Volume 310 of LNCS, pages 61–80. Springer-Verlag, Berlin, 1988.

[Felty, 1993] A. Felty. Implementing tactics and tacticals in a higher order programming language. *Journal of Automated reasoning*, 11:43–87, 1993.

[Fraenkel and Bar-Hillel, 1958] A. A. Fraenkel and Y. Bar-Hillel. *Foundations of Set Theory,* Second edition. North-Holland, Amsterdam, 1958.

[Fraenkel *et al.*, 1973] A. A. Fraenkel, Y. Bar-Hillel, and A. Levy. *Foundations of Set Theory*. North-Holland, Amsterdam, second edition, 1973.

[Fraenkel, 1922] A. A. Fraenkel. Über der begriff "definit" und die Unanhängigkeit des Auswahlaxioms. *Sitz. Berlin*, pages 253–257, 1922. English translation in [Heijenoort, 1967], 284–289.

[Friedman *et al.*, 1982] H. Friedman, K. McAloon, and S. Simpson. A finite combinatorial principles which is equivalent to the 1-consistence of predicative analysis. In G. Metakides, editor, *Logic Symposium (Patras, 1980)*, pages 197–230. North-Holland, Amsterdam, 1982.

[Friedman *et al.*, 1983] H. Friedman, S. Simpson, and R. Smith. Countable algebra and set existence axioms. *Annals of Pure and Applied Logic*, 25:141–181, 1983.

[Friedman, 1967] H. Friedman. Subsystems of set theory and analysis. Ph.D. Thesis, MIT, 1967.

[Friedman, 1969] H. Friedman. König's Lemma is weak, 1969. Mimeographed note, Stanford University.

[Friedman, 1970] H. Friedman. Iterated inductive definitions and Σ^1_2-AC. In J. Myhill *et al.*, editor, *Intuitionism and Proof Theory*, pages 435–442. North-Holland, Amsterdam, 1970.

[Friedman, 1975] H. Friedman. Some systems of second order arithmetic and their use. In *Proceedings of the International Congress of Mathematicians (Vancouver, 1974),* Volume 1, pages 235–242. Canadian Mathematical Congress, 1975.

[Friedman, 1976] H. Friedman. Systems of second order arithmetic with restricted induction, I, II (Abstracts). *Journal of Symbolic Logic*, 41:557–559, 1976.

[Friedman, 1977] H. Friedman. Set theoretic foundations for constructive analysis. *Annals of Mathematics*, 105:1–28, 1977.

[Friedman, 1978] H. Friedman. Classically and intuitionistically provable recursive functions. In G. H. Muller and D. S. Scott, editors, *Higher Set Theory*, pages 21–28. North-Holland, Amsterdam, 1978.

[Friedman, 1980] H. Friedman. A strong conservative extension of Peano arithmetic. In *The Kleene Symposium*, pages 113–122. North-Holland, 1980.

[Friedman, 1981] H. Friedman. Independence results in finite graph theory, 1981. Unpublished notes.

[Friedman, 1982] H. Friedman. Beyond Kruskal's Theorem, 1982. Unpublished notes.

[Gallier, 1990] J. Gallier. On Girard's "Candidats de Reductibilité". In P. Odifreddi, editor, *Logic and Computer Science*, pages 123–203. Academic Press, London, 1990.

[Girard *et al.*, 1989] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge University Press, Cambridge, 1989.

[Girard, 1972] J.-Y. Girard. Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre superieur, 1972. Thèse de Doctorat d'Etat.

[Glebskii *et al.*, 1969] Y. V. Glebskii, D. I. Kogan, M. I. Liogonkii, and V. A. Talanov. Range and degree of realizability of formulas in the restricted predicate calculus. *Cybernetics*, 5:142–154, 1969.

[Gloede, 1976] K. Gloede. Reflection principles and indiscernibility. In H. Müller, editor, *Sets and Classes*, pages 277–323. North-Holland, Amsterdam, 1976.

[Gödel, 1932] K. Gödel. Zur intuitionistischen Arithmetik und Zahlentheorie. *Ergebnisse eines mathematischen Kolloquiums*, 4:34–38, 1932. English translation in [Davis, 1965], pages 75–81.

[Gödel, 1940] K. Gödel. *The consistency of the axiom of choice and of the general continuum hypothesis*, volume 3 of *Annals of Mathematical Studies*. Princeton University Press, Princeton, NJ, 1940.

[Goldfarb, 1981] D. Goldfarb. The undecidability of the second order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.

[Gould, 1976] W. E. Gould. A matching procedure for ω-order logic, 1976. AFCRL Scientific Report 4.

[Gunter, 1992] C. Gunter. *Semantics of Programming Languages*. MIT Press, Cambridge, MA, 1992.

[Gurevich and Shelah, 1986] Y. Gurevich and S. Shelah. Fixed-point extensions of first-order logic. *Annals of Pure and Applied Logic*, 32:265–280, 1986.

[Gurevich, 1984] Y. Gurevich. Toward logic tailored for computational complexity. In M. M. Richter et al., editor, *Computation and Proof Theory*, Volume 1104 of LNM, pages 175–216. Springer-Verlag, Berlin, 1984.

[Gurevich, 1985] Y. Gurevich. Monadic second-order theories. In J. Barwise and S. Feferman, editors, *Model-Theoretical Logics*, pages 479–506. Springer-Verlag, Berlin, 1985.

[Gurevich, 1987] Y. Gurevich. Logic and the challenge of computer science. In E. Börger, editor, *Current Trends in Theoretical Computer Science*. Computer Science Press, 1987.

[Guttag, 1980] J. Guttag. Notes on type abstraction. *IEEE Trans. on Software Engineering SE*, 6/1:13–23, 1980.

[Hacking, 1979] I. Hacking. What is logicΓ *Journal of Philosophy*, 76:285–319, 1979.

[Hannan and Miller, 1988a] J. Hannan and D. Miller. Enriching a metalanguage with higher-order features. In J. Lloyd, editor, *Proceedings of the Workshop on Meta-Programming in Logic Programming*, Bristol, 1988. University of Bristol.

[Hannan and Miller, 1988b] J. Hannan and D. Miller. Uses of higher-order unification for implementing program transformers. In *Logic Programming: Proceedings of the Fifth International Conference and Symposium*, volume 2, pages 942–959. MIT Press, Cambridge, MA, 1988.

[Harel, 1979] David Harel. Characterizing second order logic with first order quantifiers. *Zetisch. für Math. Logik u. Grundlagen der Mathematik*, 25:419–422, 1979.

[Harper *et al.*, 1987] R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. In *Symposium on Logic in Computer Science*, pages 194–204. IEEE Computer Society Press, Washington, 1987.

[Harper *et al.*, 1989] R. Harper, D. Sannella, and A. Tarlecki. Logic representation. In *Proceedings of the Workshop on Category Theory and Computer Science*. Springer-Verlag, Berlin, 1989.

[Hazen, 1983] A. P. Hazen. Ramified type theories. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic,* Volume I. Reidel, Dordrecht, 1983.

[Hazen, 1985] A. P. Hazen. Nominalism and abstract entities. *Analysis*, 45:65–68, 1985.

[Hazen, 1989] A.P. Hazen. Two logical reductions. Preprint 2/89, October 1989. Department of Philosophy, University of Melbourne.

[Hazen, 1992] A. P. Hazen. Interpretability of Robinson Arithmetic in the

ramified second-order theory of dense linear order. *Notre Dame Journal of Formal Logic*, 33:101–111, 1992.

[Heijenoort, 1967] J.van Heijenoort. *From Frege to Gödel, A Source Book in Mathematical Logic, 1879–1931.* Harvard University Press, Cambridge, MA, 1967.

[Henkin *et al.*, 1971] L. Henkin, J. D. Monk, and A. Tarski. *Cylindric Algebras I.* North-Holland, Amsterdam, 1971.

[Henkin, 1950] Leon Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15:81–91, 1950.

[Henkin, 1961] L. Henkin. Some remarks on infinitely long formulas. In *Infinitistic Methods*, pages 167–183, Warsaw, 1961.

[Hilbert and Ackermann, 1928] D. Hilbert and W. Ackermann. *Grundzüge der theoretischen Logik.* Springer-Verlag, Berlin, 1928. English translation in *Principles of Mathematical Logic.* Chelsea, New York, 1950.

[Hilbert and Bernays, 1939] D. Hilbert and P. Bernays. *Grundlagen der Mathematik II.* Springer-Verlag, Berlin, 1939. Second edition: Springer-Verlag, 1970.

[Hintikka, 1955] J. Hintikka. Reductions in the theory of types. *Acta Philosophica Fennica*, 8:61–115, 1955.

[Hitchcock and Park, 1973] P. Hitchcock and D. Park. Induction rules and termination proofs. In *Proceedings of the First International Colloquium on Automata, Languages, and Programming*, pages 225–251. North-Holland, Amsterdam, 1973.

[Hodges, 1993] W. Hodges. *Model Theory.* Cambridge University Press, 1993.

[Hopcroft and Ullman, 1979] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA, 1979.

[Howard, 1980] W. A. Howard. The formulae–as–types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, New York, 1980. Preliminary manuscript: 1969.

[Howe, 1988] D. J. Howe. Computational metatheory in Nuprl. In E. Lusk and R. Overbeek, editors, *Ninth International Conference on Automated Deduction*, Volume 310 of LNCS, pages 238–257. Springer-Verlag, Berlin, 1988.

[Huet and Lang, 1978] G. Huet and B. Lang. Proving and applying program transformations expressed with second-order patterns. *Acta Informatica*, 11:31–55, 1978.

[Huet, 1975] G. Huet. A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[Huet, 1986] G. Huet. Formal structures for computation and deduction, 1986. Lecture Notes, Carnegie Mellon University.

[Immerman, 1986] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986. Preliminary report in *Fourteenth ACM Symposium on Theory of Computing*, 1982, pp. 147–152.

[Immerman, 1987] N. Immerman. Languages which capture complexity classes. *SIAM Journal of Computing*, 16:760–778, 1987.

[Jensen and Pietrzykowski, 1976] D. C. Jensen and T. Pietrzykowski. Mechanizing ω order type theory through unification. *Theoretical Computer Science*, 3:123–171, 1976.

[Jones and Selman, 1974] N. G. Jones and A. L. Selman. Turing machines and the spectra of first order formulas. *Journal of Symbolic Logic*, 39:139–150, 1974.

[Kechris and Moschovakis, 1977] A. Kechris and Y. Moschovakis. Recursion in higher types, 1977. In [Barwise, 1977], 681–738.

[Kelley, 1955] J. L. Kelley. *General Topology.* Van Nostrand, New York, 1955.

[Kleene, 1952] S. C. Kleene. *Introduction to Metamathematics.* Wolters-Noordhof, Groningen, 1952.

[Kleene, 1955] S. C. Kleene. On the forms of the predicates in the theory of constructive ordinals (second paper). *Amer. Journal of Math.*, 77:405–428, 1955.

[Knoblock and Constable, 1986] T. B. Knoblock and R. L. Constable. Formalized metareasoning in type theory. In *First Annual Symposium on Logic in Computer Science*, pages 237–248, Cambridge, MA, 1986. IEEE Computer Society Press.

[Kolaitis and Vardi, 1987] P. Kolaitis and M. Vardi. The decision problem for the probabilities of higher-order properties. In *Proceedings of the Nineteenth ACM Symp. on Theory of Computing*, pages 425–435. ACM, New York, 1987.

[Kolaitis and Vardi, 1990] P. Kolaitis and M. Vardi. 0-1 laws for infinitary logics. In *Proceedings of the Fifth IEEE Symp. on Logic in Computer Science*, pages 156–167, Los Alamitos, 1990. IEEE Computer Science Press.

[Kolaitis and Vardi, 1992] P. Kolaitis and M. Vardi. 0-1 laws for fragments of second-order logic: An overview. In Y. Moschovakis, editor, *Logic from Computer Science*, pages 51–72. Springer-Verlag, New York, 1992.

[Kolmogorov, 1925] A. Kolmogorov. Sur le principe de tertium non datur. *Recueil Math. de la Soc. Math. de Moscou*, 32:647–667, 1925. English translation in [Heijenoort, 1967], 414–437.

[Kozen and Parikh, 1983] D. Kozen and R. Parikh. A decision procedure

for the propositional μ-calculus. In *Proc. Workshop on Logics of Programs 1983*, Volume 164 of LNCS, pages 313–325, Berlin, 1983. Springer-Verlag.

[Kozen, 1983] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983. Preliminary version in *Proceedings of the Ninth International Colloquium on Automata, Languages, and Programming*, July 1982.

[Kozen, 1988] D. Kozen. A finite model theorem for the propositional μ-calculus. *Studia Logica*, 47:233–241, 1988. Preliminary version as Technical Report IBM RC-10320, January 1984.

[Krajíček, 1989] J. Krajíček. On the number of steps in proofs. *Annals of Pure and Applied Logic*, 41:153–178, 1989.

[Kreisel and Krivine, 1964] G. Kreisel and J.-L. Krivine. *Élements de Logique Mathématique*. Dunod, Paris, 1964. English translation: *Elements of Mathematical Logic*. North-Holland, Amsterdam, 1967.

[Kreisel and Levy, 1968] G. Kreisel and A. Levy. Reflection principles and their use for establishing the complexity of axiomatic systems. *Zetisch. für Math. Logik u. Grundlagen der Mathematik*, 14:97–191, 1968.

[Kreisel, 1963] G. Kreisel. Lectures on proof theory, 1963. Mimeographed notes, Stanford University.

[Kreisel, 1965] G. Kreisel. Mathematical logic. In T. Saaty, editor, *Lectures on Modern Mathematics*, volume III, pages 95–195. John Wiley, New York, 1965.

[Kreisel, 1967] G. Kreisel. Informal rigor and completeness proofs. In I. Lakatos, editor, *Problems in the Philosophy of Mathematics*, pages 138–186. North-Holland, Amsterdam, 1967.

[Kreisel, 1968] G. Kreisel. A survey of proof theory. *Journal of Symbolic Logic*, 33:321–388, 1968.

[Kreisel, 1971] G. Kreisel. A survey of proof theory II. In J. E. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*, pages 109–170. North-Holland, Amsterdam, 1971.

[Krynicki and Mostowski, 1994] M. Krynicki and M. Mostowski. Henkin quantifiers. In M. Krynicki *et al.*, editor, *Quantifiers*. Kluwer, 1994.

[Kuper and Vardi, 1984] G. M. Kuper and M. Vardi. A new approach to database logic. In *Proceedings of the Third ACM Conference on Principles of Database Systems*, pages 86–96, Providence, RI, 1984. ACM Press.

[Lakshman and Reddy, 1991] T. K. Lakshman and O. Reddy. Typed prolog: A semantic reconstruction of the Mycroft-O'Keefe true system. In V. Saraswat and K. Veda, editors, *Logic Programming: Proceedings of the 1991 International Symposium*, pages 202–217, Cambridge, MA, 1991. MIT Press.

[Laüchli, 1968] H. Laüchli. A decision procedure for the weak second order theory of linear order. In *Contributions to Mathematical Logic*. North-Holland, Amsterdam, 1968.

[Leblanc, 1976] H. Leblanc. *Truth-Value Semantics*. North-Holland, Amsterdam, 1976.

[Lee and Pleban, 1987] P. Lee and U. F. Pleban. A realistic compiler generator based on high-level semantics. In *Proceedings of the Fourteenth Annual ACM Symposium on Principles of Programming Languages*, pages 284–295. ACM Press, New York, 1987.

[Leivant, 1983] D. Leivant. Reasoning about functional programs and complexity classes associated with type disciplines. In *Twenty-fourth Annual Symposium on Foundations of Computer Science*, pages 460–469, 1983.

[Leivant, 1985] D. Leivant. Syntactic translations and provably recursive functions. *Journal of Symbolic Logic*, 50:682–688, 1985.

[Leivant, 1987] D. Leivant. Descriptive characterizations of computational complexity. In *Second Annual Conference on Structure in Complexity Theory*, pages 203–217, Washington, 1987. IEEE Computer Society Press. Revised in *Journal of Computer and System Sciences*, 39:51–83, 1989.

[Leivant, 1990a] D. Leivant. Computationally based set existence principles. In W. Sieg, editor, *Logic and Computation*, Volume 106 of Contemporary Mathematics, pages 197–211. American Mathematical Society, Providence, RI, 1990.

[Leivant, 1990b] D. Leivant. Contracting proofs to programs. In P. Odifreddi, editor, *Logic and Computer Science*, pages 279–327. Academic Press, London, 1990.

[Leivant, 1990c] D. Leivant. Inductive definitions over finite structures. *Information and Computation*, 89:95–108, 1990.

[Leivant, 1991a] D. Leivant. A foundational delineation of computational feasiblity. In *Proceedings of the Sixth IEEE Conference on Logic in Computer Science (Amsterdam)*, Washington, 1991. IEEE Computer Society Press.

[Leivant, 1991b] D. Leivant. Semantic characterization of number theories. In Y. Moschovakis, editor, *Logic from Computer Science*, pages 295–318. Springer-Verlag, New York, 1991.

[Leivant, 1994] D. Leivant. A foundational delineation of poly-time, 1994. To appear in *Information and Computation*.

[Levy, 1960] A. Levy. Principles of reflection in axiomatic set theory. *Fundamenta Math.*, 49:1–10, 1960.

[Lindström, 1969] P. Lindström. On extensions of elementary logic. *Theoria*, 35:1–11, 1969.

[Livchak, 1983] A. Livchak. The relational model for process control. *Automatic Documentation and Mathematical Linguistics*, 4:27–29, 1983.

[Löb, 1976] M. H. Löb. Embedding first order predicate logic in fragments of intuitionistic logic. *Journal of Symbolic Logic*, 41:705–718, 1976.

[Löwenheim, 1915] L. Löwenheim. Über Möglichkeiten im Relativkalkül. *Math. Ann.*, 76, 1915.

[Lubarsky, 1993] R. Lubarsky. μ-definable sets of integers. *Journal of Symbolic Logic*, 58:291–313, 1993.

[Makinouchi, 1977] A. Makinouchi. A consideration on normal form of not necessarily normalized relation in the relational data model. In *Proc. Third Symp. on VLDB*, pages 447–453, 1977.

[Marshall, 1989] M. Victoria Marshall. Higher order reflection principles. *Journal of Symbolic Logic*, 54:474–489, 1989.

[Martin-Löf, 1973a] P. Martin-Löf. Hauptsatz for intuitionistic simple type theory. In P. Suppes, L. Henkin, A. Joja, and G. C. Moisil, editors, *Logic, Methodology and Philosophy of Science IV*, pages 279–290, Amsterdam, 1973. North-Holland.

[Martin-Löf, 1973b] P. Martin-Löf. An intuitionistic theory of types: Predicative part. In H. E. Rose and J. C. Shepherdson, editors, *Logic Colloquium '73*, pages 73–118, Amsterdam, 1973. North-Holland.

[Martin-Löf, 1980] P. Martin-Löf. Constructive mathematics and computer programming. In *Logic, Methodology and Philosophy of Science VI*, pages 153–175. North-Holland, Amsterdam, 1980.

[Martin, 1975] D. Martin. Borel determinacy. *Annals of Math.*, 102:363–371, 1975.

[Mendelson, 1964] E. Mendelson. *Introduction to Mathematical Logic*. Van Nostrand, Princeton, 1964.

[Miller and Nadathur, 1986] D. A. Miller and G. Nadathur. Higher-order logic programming. In *Proceedings of the Third International Conference on Logic Programming*. Springer-Verlag, Berlin, 1986.

[Miller *et al.*, 1991] D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[Miller, 1983] D. Miller. Proofs in higher-order logic. Ph.D. Thesis, Carnegie-Mellon University, 1983.

[Miller, 1987] D. Miller. A compact representation of proofs. *Studia Logica*, pages 347–370, 1987.

[Miller, 1988] D. Miller. Logic programming based on higher-order hereditary harrop formulas, 1988. Technical Report (publishedΓ).

[Miller, 1989] D. Miller. Abstractions in logic programming. In P. Odifreddi, editor, *Logic and Computer Science*. Academic Press, New York, 1989.

[Miller, 1993] D. Miller. Higher order logic programming. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming,* Volume 5. Oxford University Press, Oxford, 1993.

[Milner and Tofte, 1991] R. Milner and M. Tofte. Co-induction and relational semantics. *Theoretical Computer Science*, 87:209–220, 1991.

[Mitchell, 1990] J. C. Mitchell. Types systems for programming languages. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 365–458. North-Holland, Amsterdam, 1990.

[Monk, 1976] J. D. Monk. *Mathematical Logic*. Springer-Verlag, New York, 1976.

[Montague, 1965a] R. Montague. Reductions of higher-order logic. In J.W. Addison, L. Henkin, and A. Tarski, editors, *The Theory of Models*, pages 251–264. North-Holland, Amsterdam, 1965.

[Montague, 1965b] R. Montague. Set theory and higher order logic. In J. Crossley and M. Dummett, editors, *Formal Systems and Recursive Functions*, pages 131–148. North-Holland, Amsterdam, 1965.

[Morse, 1965] A. P. Morse. *A Theory of Sets*. Academic Press, New York, 1965.

[Moschovakis, 1974] Y. Moschovakis. *Elementary Induction on Abstract Structures*. North-Holland, Amsterdam, 1974.

[Moschovakis, 1993] Y. Moschovakis. Sense and denotation as algorithm and value. In *Proceedings of the 1993 ASL Summer Meeting in Helsinki*, Lecture Notes in Logic. Springer-Verlag, 1993. To appear.

[Mosses, 1990] P. D. Mosses. Denotational semantics. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 577–632. North-Holland, Amsterdam, 1990.

[Mostowski, 1939] A. Mostowski. Über die Unabhängigkeit des Wohlordungssatzes vom Ordungprinzip. *Fund. Math.*, 32:201–252, 1939.

[Mostowski, 1951] A. Mostowski. Some impredicative definitions in the axiomatic set theory. *Fundamenta Math.*, 37:111–124, 1951.

[Mostowski, 1952] A. Mostowski. *Sentences Undecidable in Formalized Arithmetic: An Exposition of the Theory of Kurt Gödel*. North-Holland, Amsterdam, 1952.

[Mostowski, 1968] A. Mostowski. Craig's interpolation theorem in some extended systems of logic. In B. van Rootselaar and J. F. Staal, editors, *Proceedings of the Third International Conference on Logic, Methodology and Philosophy of Science*, pages 87–103. North-Holland, Amsterdam, 1968.

[Mundici, 1985] D. Mundici. Other quantifiers: An overview. In J. Barwise and S. Feferman, editors, *Model Theoretic Logic*, pages 211–233. North-Holland, Amsterdam, 1985.

[Mycroft and O'Keef, 1984] A. Mycroft and R. A. O'Keef. A polymorphic type system for prolog. *Artificial Intelligence*, pages 295–307, 1984.

[Nadathur and Miller, 1988] G. Nadathur and D. Miller. An overview of λ-prolog. In K. A. Bowen and R. A. Kowalski, editors, *Fifth International Logic Programming Conference*, pages 810–827. MIT Press, Cambridge, MA, 1988.

[Nadathur and Miller, 1990] G. Nadathur and D. Miller. Higher-order horn clauses. *Journal of the ACM*, 37:777–814, 1990.

[Nadathur, 1987] G. Nadathur. A higher-order logic as the basis for logic programming. Ph.D. Thesis, University of Pennsylvania, 1987.

[Neumann, 1925] J. von Neumann. Eine axiomatisierung der mengelehre. *J. f. Math.*, 154:219–240, 1925. Corrections *J. f. Math.*, 155:128, 1926. English translation in [Heijenoort, 1967], 393–413.

[Novak, 1951] I. Novak. A construction for models of consistent systems. *Fundamenta Math.*, 37:87–110, 1951.

[Parikh, 1973] R. J. Parikh. Some results on the length of proofs. *Trans. Amer. Math. Soc.*, 177:29–36, 1973.

[Parikh, 1986] R. J. Parikh. Introductory note to 1936(a). In *Kurt Gödel, Collected Works,* Volume 1, pages 394–397. Oxford University Press, Oxford, 1986.

[Paris and Harrington, 1977] J. Paris and L. Harrington. A mathematical incompleteness in Peano Arithmetic, 1977. In [Barwise, 1977, pp. 1133–1142].

[Park, 1970] D. Park. Fixpoint induction and proof of program semantics. In Meltzer and Michie, editors, *Machine Intelligence V*, pages 59–78. Edinburgh University Press, Edinburgh, 1970.

[Paulin-Mohring, 1989a] C. Paulin-Mohring. Extracting F_ω programs from proofs in the calculus of constructions. In *Sixteenth Annual Symposium on Principles of Programming Languages*, pages 89–104. ACM Press, New York, 1989.

[Paulin-Mohring, 1989b] C. Paulin-Mohring. Extraction de programmes dans le Calcul des Constructions. Ph.D. Thesis, Thèse, Université de Paris VII, 1989.

[Paulson, 1987] L. C. Paulson. *Logic and Computation: Interactive Proof with Cambridge LCF.* Cambridge University Press, 1987.

[Paulson, 1989] L. C. Paulson. The foundation of a generic theorem prover. *Journal of Automated Reasoning*, 5:363–397, 1989.

[Paulson, 1990] L. C. Paulson. Isabelle: The next 700 theorem provers. In P. Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press, New York, 1990.

[Petersson, 1982] K. Petersson. A programming system for type theory. Technical Report 21, Department of Computer Science, Chalmers Uni-

versity, Göteborg, 1982.

[Pfenning and Rohwedder, 1991] F. Pfenning and E. Rohwedder. Implementing the meta-theory of deductive systems, 1991. Preprint.

[Pfenning, 1989] F. Pfenning. Elf: A language for logic definition and verified meta-programming. In *Fourth Annual Symposium on Logic in Computer Science*, pages 313–322, Washington, 1989. IEEE Computer Society Press.

[Pitts, 1993] A. M. Pitts. A co-induction principle for recursively defined domains. *Theoretical Computer Science*, 1993. To appear.

[Poincaré, 1902] H. Poincaré. Du rôle de l'intuition et de la logique en mathématiques. *C.R. du Deuxième Congrès Int. des Math., Paris 1900*, pages 200–202, 1902.

[Poincaré, 1910] H. Poincaré. Über transfinite Zahlen. In *Sechs Vorträge über ausgewahlte Gegenstände aus der reiner Mathematik und mathematischen Physik*, pages 43–48, Leipzig, 1910.

[Prawitz, 1965] D. Prawitz. *Natural Deduction.* Almqvist and Wiksell, Uppsala, 1965.

[Prawitz, 1968] D. Prawitz. Hauptsatz for higher order logic. *Journal of Symbolic Logic*, 33:452–457, 1968.

[Prawitz, 1972] D. Prawitz. Ideas and results in proof theory. In J. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*, pages 235–307. North-Holland, Amsterdam, 1972.

[Quine, 1937] W. V. O. Quine. New foundations for mathematical logic. *Amer. Math. Monthly*, 44:80–101, 1937.

[Quine, 1951] W. V. O. Quine. *Mathematical Logic,* Revised edition. MIT Press, Cambridge, MA, 1951. (First edition: 1940).

[Quine, 1970] W. V. O. Quine. *Philosophy of Logic.* Prentice-Hall, Englewood Cliffs, NJ, 1970.

[Rabin, 1961] M. Rabin. Non-standard models and independence of the induction axiom. In *Essays in the Foundations of Mathematics*, pages 287–299. The Magnes Press, Jerusalem, 1961.

[Rabin, 1969] M. Rabin. Decidability of second order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969.

[Reynolds, 1972] J. C. Reynolds. Definitional interpreters for higher-order programming languages. In *Proceedings of the 25th ACM National Conference*, pages 717–740, New York, 1972. ACM Press.

[Reynolds, 1974] J. C. Reynolds. Towards a theory of type structure. In *Proc. Colloque sur la Programmation*, Volume 19 of LNCS, pages 408–425. Springer-Verlag, Berlin, 1974.

[Reynolds, 1981] J. C. Reynolds. The essence of Algol. In J. W. de Bakker and J. C. van Vliet, editors, *Algorithmic Languages*, pages 345–372. North-Holland, Amsterdam, 1981.

[Reynolds, 1988] J. C. Reynolds. Preliminary design of the programming language Forsythe. Technical Report CMU-CS-88-159, Carnegie Mellon University, 1988.

[Robinson, 1965] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12:23–41, 1965.

[Robinson, 1969] J. A. Robinson. Mechanizing higher order logic. *Machine Intelligence*, 4:151–170, 1969.

[Rogers, 1967] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York, 1967.

[Rosse and Wang, 1950] J. R. Rosse and A. Wang. Nonstandard models for formal logic. *Journal of Symbolic Logic*, 15:113–129, 1950.

[Rougemont, 1987] M. de Rougemont. Second order and inductive definability on finite structures. *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 33:47–63, 1987.

[Rubin and Rubin, 1966] H. Rubin and J. E. Rubin. *Equivalents of the Axiom of Choice*. North-Holland, Amsterdam, 1966.

[Russell, 1908] B. Russell. Mathematical logic as based on the theory of types. *Amer. J. of Math.*, 30:222–262, 1908.

[Ryll-Nardzewski, 1953] C. Ryll-Nardzewski. The role of the axiom of induction in elementary arithmetic. *Fundamenta Mathematicae*, 39:239–263, 1953.

[Scedrov, 1990] A. Scedrov. A guide to polymorphic types. In P. Odifreddi, editor, *Logic and Computer Science*, pages 387–420. Academic Press, London, 1990.

[Schönfinkel, 1924] M. Schönfinkel. Über die Bausteine der mathematischen Logik. *Mathematische Annalen*, 92:305–316, 1924. English translation: On the building blocks of mathematical logic, in [Heijenoort, 1967], 355-366.

[Schütte, 1951] K. Schütte. Beweistheoretische Erfassung der unendlischen Induktion in der reinen Zahlentheorie. *Math. Annalen*, 122:369–389, 1951.

[Schütte, 1952] K. Schütte. Beweistheoretische Untersuchung der versweigten Analysis. *Math. Annalen*, 124:123–147, 1952.

[Schütte, 1960a] K. Schütte. *Beweistheorie*. Volume 103 of Grundlagen der mathematischen Wissenschaften. Springer-Verlag, Berlin, 1960. English translation: *Proof Theory*, Springer-Verlag, Berlin, 1977.

[Schütte, 1960b] K. Schütte. Syntactic and semantic properties of simple type theory. *Journal of Symbolic Logic*, 25:305–325, 1960.

[Schütte, 1965a] K. Schütte. Eine Grenze für die Beweisbarkeit der transfiniten Induktion in der verzweigten Typenlogik. *Archiv für math. Logik und Grundlagenforschung*, 7:45–60, 1965.

[Schütte, 1965b] K. Schütte. Predicative well orderings. In J. Crossley and M. Dummett, editors, *Formal Systems and Recursive Functions*, pages 280–303. North-Holland, Amsterdam, 1965.

[Schwarts *et al.*, 1986] J. T. Schwarts, R. B. K. Dewar, E. Dubinsky, and E. Schonberg. *Programming with Sets: An Introduction to SETL.* Springer-Verlag, New York, 1986.

[Schwichtenberg, 1977] H. Schwichtenberg. Proof theory: Some applications of cut-elimination, 1977. In [Barwise, 1977], pages 867–895.

[Scott and de Bakker, 1969] D. Scott and J.W. de Bakker. A theory of programs. Unpublished manuscript. IBM, Vienna, 1969.

[Scott and Strachey, 1971] D. S. Scott and C. Strachey. Towards a mathematical semantics for computer languages. In J. Fox, editor, *Computers and Automata*, pages 19–46. Brooklyn Polytechnic Institute Press, 1971.

[Shapiro, 1991] S. Shapiro. *Foundation without Foundationalism: A Case for Second-Order Logic.* Oxford University Press, Oxford, 1991.

[Sieg, 1985] W. Sieg. Fragments of arithmetic. *Annals of Pure and Applied Logic*, 28:33–71, 1985.

[Sieg, 1987] W. Sieg. Provably recursive functionals of theories with König's Lemma. *Rend. Sem. Mat. Univers. Politecn. Torino, Fasc. Specale: Logic and Computer Science*, pages 75–92, 1987.

[Sieg, 1990] W. Sieg. Review of [Simpson, 1985a]. *Journal of Symbolic Logic*, 55:870–874, 1990.

[Simpson, 1982a] S. Simpson. Σ^1_1 and Π^1_1 transfinite induction. In D. Van Dalen et al., editor, *Logic Colloquium '80*, pages 239–253. North Holland, 1982.

[Simpson, 1982b] S. Simpson. Set theoretic aspects of ATR_0. In D. Van Dalen et als, editor, *Logic Colloquium '80*, pages 254–271. North Holland, 1982.

[Simpson, 1984] S. Simpson. Which set existence axioms are needed to prove the Cauchy/Peano Theorem for ordinary differential equationsΓ *Journal of Symbolic Logic*, 49:783–802, 1984.

[Simpson, 1985a] S. Simpson. Friedman's research on subsystems of second order arithmetic. In L. A. Harrington et al., editor, *Harvey Friedman's Research on the Foundations of Mathematics*, pages 137–159. North-Holland, Amsterdam, 1985.

[Simpson, 1985b] S. Simpson. Nonprovability of certain combinatorial properties of finite trees. In L. A. Harrington et al., editor, *Harvey Friedman's Research on the Foundations of Mathematics*, pages 87–117. North-Holland, Amsterdam, 1985.

[Simpson, 1986] S. Simpson. Subsystems of Z_2 and Reverse Mathematics. Appendix to: G. Takeuti, *Proof Theory*, pages 434–448, 1986. Second edition. North-Holland, Amsterdam.

[Skolem, 1919] T. Skolem. Unterschungen über die Axiome des Klassenkalküls und über Produktation- und Summationsprobleme. *Vid. Skrifter I, Mat.-nat. Klasse*, 3, 1919.

[Smorynski, 1977] C. Smorynski. The incompleteness theorems, 1977. In [Barwise, 1977], 821–865.

[Solovay *et al.*, 1978] R. Solovay, W. Reinhardt, and A. Kanamori. Strong axioms of infinity and elementary embeddings. *Annals of Math. Logic*, 13:73–116, 1978.

[Spector, 1962] C. Spector. Provable recursive functions of analysis. In J. Dekker, editor, *Recursive Function Theory*, volume 5 of *Proceedings of Symposia in Pure Mathematics*, pages 1–27, Providence, 1962. American Mathematical Society.

[Spencer and Shelah, 1987] J. Spencer and S. Shelah. Threshold spectra for random graphs. In *Proceedings of the Nineteenth ACM Symp. on Theory of Computing*, pages 421–424, New York, 1987. ACM.

[Spencer, 1993] J. Spencer. Zero-one laws with variable probability. *Journal of Symbolic Logic*, 58:1–14, 1993.

[Statman, 1978] R. Statman. Bounds for proof-search and speed-up in the predicate calculus. *Annals of Mathematical Logic*, 15:225–287, 1978.

[Steel, 1976] J. R. Steel. Determinateness and subsystems of analysis. Ph.D. Thesis, University of California at Berkeley, 1976.

[Stockmeyer, 1974] L.J. Stockmeyer. The complexity of decision problems in automata theory and logic. Technical Report MAC TR-133, MIT, Cambridge, MA, 1974.

[Sundholm, 1981] G. Sundholm. Hacking's logic. *Journal of Philosophy*, 78:160–168, 1981.

[Tait, 1966] W. Tait. A non-constructive proof of Gentzen's Hauptsatz for second order predicate logic. *Bull. Amer. Math. Soc.*, 72:980–983, 1966.

[Takahashi, 1967] M. Takahashi. A proof of cut-elimination theorem in simple type-theory. *Journal of the Math. Soc. of Japan*, 19:399–410, 1967.

[Takeuti, 1975] G. Takeuti. *Proof Theory*. North-Holland, Amsterdam, 1975.

[Tarski, 1952] A. Tarski. Some notions and methods on the borderline of algebra and metamathematics. In *Proceedings of the International Congress of Mathematicians (1950)*, volume 1, pages 705–720, Providence, 1952. American Mathematical Society.

[Tennent, 1991] R. D. Tennent. *Semantics of Programming Languages*. Prentice-Hall, Englewood Cliffs, NJ, 1991.

[Tharp, 1975] L. Tharp. Which logic is the right logicΓ *Synthese*, 31:1–31, 1975.

[Trakhtenbrot, 1950] B. Trakhtenbrot. The impossibility of an algorithm for the decision problem for finite domains. *Doklady Akademii Nauk SSSR*, 70:569–572, 1950. In Russian.

[Troelstra, 1973] A. S. Troelstra. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis.* Volume 344 of LNM. Springer-Verlag, Berlin, 1973.

[Vardi, 1982] M. Vardi. Complexity and relational query languages. In *Fourteenth Symposium on Theory of Computing*, pages 137–146. ACM, New York, 1982.

[Vopénka and Hájek, 1972] P. Vopénka and P. Hájek. *The Theory of Semisets.* North-Holland, Amsterdam, 1972.

[Walkoe, 1970] W. Walkoe. Finite partially ordered quantification. *Journal of Symbolic Logic*, 35:535–555, 1970.

[Wang, 1949] H. Wang. On Zermello's and von Neumann's axioms for set theory. *Proc. Academy of Sciences*, 35:150–155, 1949.

[Wang, 1954] H. Wang. The formalization of mathematics. *Journal of Symbolic Logic*, 19:241–266, 1954.

[Wang, 1962] H. Wang. *Some formal details on predicative set theories*, chapter XXIV. Science Press, Peking, 1962. Republished in 1964 by North Holland, Amsterdam. Republished in 1970 under the title *Logic, Computers, and Sets* by Chelsea, New York.

[Weber, 1991] M. Weber. *A Meta-Calculus for Formal System Development.* Oldenbourg Verlag, München, 1991.

[Weyl, 1918] H. Weyl. *Das Kontinuum: Kritische Untersuschungen über die Grundlagen der Analysis.* Veit, Leipzig, 1918. Reprinted in: H. Weyl, E. Landau, and B. Riemann, *Das Kontinuum und andere Monographie*, Chelsea, New York, 1960.

[Whitehead and Russell, 1929] A. N. Whitehead and B. Russell. *Principia Mathematicae.* Cambridge University Press, second edition, 1929.

[Yardeni *et al.*, 1991] E. Yardeni, T. Fruehwirth, and E. Shapiro. Polymorphically typed logic programs. In F. Pfenning, editor, *Types in Logic Programming.* MIT Press, Cambridge, MA, 1991.

[Zahn, 1978] P. Zahn. *Ein Konstruktiver Weg zur Masstheorie und Funktionalanalysis.* Wissenschaftlische Buchgeselschaft, 1978.

[Zermelo, 1908] E. Zermelo. Untersuchungen über die Grundlagen der Mengenlehr I. *Math. Annalen*, 65:261–281, 1908. English translation in [Heijenoort, 1967], 199–215.

[Zermelo, 1930] E. Zermelo. Über Grenzzahlen und Mengenbereiche. *Fund. Math.*, 16:129–47, 1930.

[Barwise, 1977] [Davis, 1965] [Heijenoort, 1967]