Lexicographic Path Induction

Jeffrey Sarnat[1] and Carsten Schürmann[2]

[1] Yale University
jeffrey.sarnat@yale.edu
[2] IT University of Copenhagen
carsten@itu.dk

Abstract. Programming languages theory is full of problems that reduce to proving the consistency of a logic, such as the normalization of typed lambda-calculi, the decidability of equality in type theory, equivalence testing of traces in security, etc. Although the principle of transfinite induction is routinely employed by logicians in proving such theorems, it is rarely used by programming languages researchers who often prefer alternatives such as proofs by logical relations and model theoretic constructions. In this paper we harness the well-foundedness of the lexicographic path ordering to derive an induction principle that combines the comfort of structural induction with the expressive strength of transfinite induction. Using lexicographic path induction, we give a consistency proof of Martin-Löf's intuitionistic theory of inductive definitions. The consistency of Heyting arithmetic follows directly, and weak normalization for Gödel's T follows indirectly; both have been formalized in a prototypical extension of Twelf.

1 Introduction

Programming languages theory is full of problems that reduce to proving the consistency of a logic, such as the normalization of typed λ-calculi, the decidability of equality in type theory, equivalence testing of traces in security, etc. Although the principle of transfinite induction is routinely employed by logicians in proving such theorems, it is rarely used by programming languages researchers who often prefer alternatives such as proofs by logical relations and model theoretic constructions.

This phenomenon can be explained at least in part by the fact that ordinals can be notoriously tricky to work with. The Burali-Forti paradox illustrates that any ordinal notation system is necessarily incomplete, and in practice, as ordinals get bigger, the notation systems needed to describe them become more complex.

In contrast, the lexicographic path ordering (LPO) is both powerful (its order type approaches the small Veblen ordinal) and well understood by computer scientists. Furthermore, it is easy to implement and has been used to prove the termination of term rewriting systems (TRSs) for decades, where, ironically, its considerable proof-theoretic strength cannot be fully harnessed.

For many logical systems, consistency follows directly from the termination of a cut-elimination procedure. Since the LPO is more than strong enough to prove

the consistency of arithmetic, and cut-elimination procedures can be expressed as term rewriting systems, one might hope to demonstrate the consistency of arithmetic using the lexicographic path ordering to show the termination of such a TRS.

However, this is impossible. Buchholz [Buc95] has shown that if one proves the termination of a TRS using the LPO, then the proof can be modified such that it is valid in a fragment of arithmetic. By Gödel's second incompleteness theorem, one cannot prove the consistency of arithmetic from within a fragment of arithmetic, therefore a cut-elimination procedure for arithmetic cannot be shown terminating formulated as a TRS via the LPO.

In this paper, we show how to harness the strength of the LPO in form of an induction principle that we call *lexicographic path induction*, which combines the comfort of structural induction with the expressive strength of transfinite induction.

The consistency of arithmetic and weak normalization of Gödel's T are well-known examples of transfinite induction. We give a novel consistency proof by lexicographic path induction of an intuitionistic theory of inductive definitions, based on the system by Martin-Löf [ML71] that inspired the definition of inductive types in type theory [Dyb91,PM93], and several sequent calculi in the programming languages literature [MM00,MT03,Bro06,GMN08]. The consistency of Heyting Arithmetic follows as a simple corollary, and the weak normalization of Gödel's T follows via a structural logical relation [SS08]. Both have been formalized in a prototypical extension of Twelf (`http://www.twelf.org/lpo/`).

The paper is organized as follows. In the Section 2 we introduce the lexicographic path orderings and the principle of lexicographic path induction. In Section 3 we introduce a sequent calculus and for intuitionistic logic with inductive definitions. In Section 4, we prove the consistency of this logic. In Section 5, we conclude and describe related and future work.

2 The Lexicographic Path Ordering

The lexicographic path ordering (LPO) provides a modular way of specifying orderings on finite labeled trees whose constructors have fixed arity.

Given a signature Σ of fixed arity constructors, whose elements we denote generically by the letters f and g, labeled trees are defined as follows

$$\textbf{Labeled Trees } \mathsf{s}, \mathsf{t} ::= \mathsf{f}(\mathsf{s}_1, \ldots, \mathsf{s}_n)$$

where the arity of f, denoted $\#\mathsf{f}$, is n for $n \geq 0$. We use Σ^n to denote the constructors of Σ of arity n. Although signatures can in principle be infinite, all of the signatures considered in this paper are finite.

Definition 1 (Lexicographic Path Ordering). *Given a precedence relation $<$ on Σ we define $<_{\mathsf{lpo}}$ as the smallest relation on trees that satisfies the following:*
$\mathsf{s} = \mathsf{f}(\mathsf{s}_1, \ldots, \mathsf{s}_n) <_{\mathsf{lpo}} \mathsf{g}(\mathsf{t}_1, \ldots, \mathsf{t}_m) = \mathsf{t}$ *iff at least one of the following holds:*

1. $\mathsf{f} < \mathsf{g}$ *and for all* $i, \in 1 \ldots, n$ $\mathsf{s}_i <_{\mathsf{lpo}} \mathsf{t}$

2. $\mathsf{f} = \mathsf{g}$ *and there exists* $k \in 1, \ldots, n$ *s.t. for all* $i < k$ $\mathsf{s}_i = \mathsf{t}_i$, $\mathsf{s}_k < \mathsf{t}_k$ *and for all* $j \in k+1, \ldots, n$ $\mathsf{s}_j <_{\mathsf{lpo}} \mathsf{t}$
3. $\mathsf{s} \leq_{\mathsf{lpo}} \mathsf{t}_i$, *for some* $i \in 1, \ldots, n$

where $\mathsf{s} \leq_{\mathsf{lpo}} \mathsf{t}$ *is shorthand for "*$\mathsf{s} = \mathsf{t}$ *or* $\mathsf{s} <_{\mathsf{lpo}} \mathsf{t}$*."*

We are concerned exclusively with instances of $<_{\mathsf{lpo}}$ where $<$ is transitive and well-founded (and therefore irreflexive). LPOs have several nice properties, including the preservation of transitivity and well-foundedness of $<$.

Lemma 1 (Properties of LPO).

- *(Subterm)* $\mathsf{t} <_{\mathsf{lpo}} \mathsf{f}(\ldots \mathsf{t} \ldots)$
- *(Monotonicity) If* $\mathsf{s} <_{\mathsf{lpo}} \mathsf{t}$, *then* $\mathsf{f}(\ldots \mathsf{s} \ldots) <_{\mathsf{lpo}} \mathsf{f}(\ldots \mathsf{t} \ldots)$.
- *(Transitivity) If* $<$ *is transitive, then so is* $<_{\mathsf{lpo}}$.
- *(Well-foundedness) If* $<$ *is well-founded, then so is* $<_{\mathsf{lpo}}$.
- *(Big head) If* $\mathsf{s} = \mathsf{f}(\mathsf{s}_1, \ldots, \mathsf{s}_n)$, $\mathsf{g}_1 < \mathsf{f}, \ldots, \mathsf{g}_m < \mathsf{f}$ *and* t *is built up from* $\mathsf{s}_1, \ldots, \mathsf{s}_n$ *and* $\mathsf{g}_1, \ldots, \mathsf{g}_m$ *then* $\mathsf{t} <_{\mathsf{lpo}} \mathsf{s}$.

Proof. The subterm, monotonicity and transitivity properties are shown in [KL80], well-foundedness is shown in [Buc95]. The big head property can be shown by a straightforward induction on the structure of t.

Example 1

Let $\Sigma = \{\mathsf{z}, \mathsf{succ}, \mathsf{op}\}$, *where* $\#\mathsf{z} = 0$, $\#\mathsf{s} = 1$ *and* $\#\mathsf{op} = 2$, *and let* $<$ *be defined as* $\mathsf{z} < \{\mathsf{s}, \mathsf{op}\}$, $\mathsf{s} < \mathsf{op}$. *The following inequalities hold for every* s *and* t

1. $\mathsf{succ}^n(\mathsf{s}) <_{\mathsf{lpo}} \mathsf{op}(\mathsf{s}, \mathsf{t})$ *and* $\mathsf{succ}^n(\mathsf{t}) <_{\mathsf{lpo}} \mathsf{op}(\mathsf{s}, \mathsf{t})$, *for every* n
2. $\mathsf{succ}(\mathsf{op}(\mathsf{s}, \mathsf{t})) <_{\mathsf{lpo}} \mathsf{op}(\mathsf{s}, \mathsf{succ}\ \mathsf{t})$
3. $\mathsf{op}(\mathsf{s}, \mathsf{op}(\mathsf{s}, \mathsf{op}(\mathsf{s}, \mathsf{t}))) <_{\mathsf{lpo}} \mathsf{op}(\mathsf{succ}\ \mathsf{s}, \mathsf{t})$
4. $\mathsf{op}(\mathsf{s}, \mathsf{op}(\mathsf{succ}\ \mathsf{s}, \mathsf{t})) <_{\mathsf{lpo}} \mathsf{op}(\mathsf{succ}\ \mathsf{s}, \mathsf{succ}\ \mathsf{t})$

The first inequality can be seen as an instance of the big head property. The second inequality in highlights another interesting property of LPOs: if a large constructor (in this case op) occurs beneath a small constructor (in this case succ), then "bubbling up" the larger constructor results in larger term, or viewed the other way, "bubbling up" the smaller constructor results in a smaller term; this observation will play an important role in Lemma 6. The third inequality highlights the application of the second clause of Definition 1: in a sense, one can think a partially applied constructor as being a constructor in its own right, where the "precedence" of $\mathsf{op}(\mathsf{s}, -)$ is smaller than $\mathsf{op}(\mathsf{succ}\ \mathsf{s}, -)$. The last inequality can be used to help show that the Ackermann function, when formulated as a term rewriting system, terminates.

Definition 2 (Principle of Lexicographic Path Induction). *The* Principle of Lexicographic Path Induction *is the principle of well-founded induction with the LPO as the well-founded ordering.*

In the next sections, we will show one of the application of the principle, which justifies its definition, and demonstrates how one can proof the consistency of Heyting arithmetic via this principle.

3 The Intuitionistic Theory of Inductive Definitions

In the previous section, we saw that lexicographic path induction can be seen as an instance of well-founded induction. However, more conventional induction principles can be obtained from the notion of inductive definitions.

In mathematics, inductive definitions give rise to monotone operators, and the Tarski-Knaster fixed point theorem can be used to justify the existence of, and a notion of induction over, inductively defined families [Pie02]. Inductive definitions can also be primitively formalized in the language of first order logic, using a theory of iterated inductive definitions. Martin-Löf's intuitionistic theory of iterated inductive definitions ($\mathrm{IID}_{<\omega}$) [ML71] is notable for its formulation as a natural deduction calculus, where predicate symbols play the role of inductively defined families, atomic introduction rules play the role of monotone operator and atomic elimination rules play the role induction principles. Martin-Löf proved the consistency of $\mathrm{IID}_{<\omega}$ using a logical relations argument; here, we give a sequent calculus formulation of (non-iterated) intuitionistic inductive definitions (IID_0) and in Section 4 we prove its consistency using lexicographic path induction. The proof-theoretic strength of IID_0 is that of arithmetic [ML71].

3.1 Formulas and Proofs

The language of IID_0 is parameterized by a term algebra, whose fixed arity constructors we denote generically using the letter c, by some collection of fixed arity predicate symbols, whose elements we denote generically using the letter a, and some collection X of term variables, whose elements we refer to generically using the letters x and y. As a running example, we consider an instantiation of IID_0 with a term algebra corresponding to the natural numbers, i.e. the constant z and unary constructor s, and with the predicate symbols $\bot, \mathit{nat}$ and eq, whose arities are 0, 1, and 2, respectively. Formulas are defined below.

$$
\begin{array}{ll}
\textbf{Terms} & t, u ::= x \mid c(t_1, \ldots, t_n) \\
\textbf{Formulas} & F, G ::= a(t_1, \ldots, t_n) \mid F \wedge G \mid F \vee G \mid F \Rightarrow G \mid \forall x.F \mid \exists x.F \\
\textbf{Predicates} & P, Q ::= (x_1, \ldots, x_n \mapsto F)
\end{array}
$$

An n-ary predicate is the abstraction of a formula over n bound variables: note that any formula can be viewed as a 0-ary predicate, and vice-versa. We say that a formula is *atomic* if it of the form $a(t_1, \ldots, t_n)$, which we denote using the letters A and B; otherwise it is *compound*, which we denote using the letters K and L.

We write $t[t'/x]$ and $F[t/x]$ for the usual notions of capture-avoiding substitution on terms and formulas, and $P(t_1, \ldots, t_n)$ for $F[t_1/x_1] \ldots [t_n/x_n]$ when $P = x_1, \ldots, x_n \mapsto F$.

We formalize the notion of provability using sequents of the form $\Gamma \vdash F$, where Γ is the notion of context defined below. We depart slightly from convention by using Γ to keep track of not only which logical hypotheses may be used freely in the deduction of a sequent, but also which free term variables may be used in the deduction as well. Although naming the logical hypotheses

in Γ is not useful when presenting IID_0's proof rules, doing so is useful in the presentation of IID_0's proof terms, which will be introduced in Section 3.2. To this end, we assume that we are given a set of hypothetical variables H, whose elements we denote generically using the letter h.

$$\textbf{Contexts } \Gamma ::= \cdot \mid \Gamma, h{:}A \mid \Gamma, x$$

In order for Γ to be well formed, we require that each element of X or H occur in Γ at most once; this condition can typically be satisfied by the implicit renaming of bound variables. We write $\Gamma[t/x]$ for the capture-avoiding substitution of all occurrences of x in Γ with t (if x is an entry in Γ, it is deleted). The relation $\Gamma \geq \Gamma'$ holds whenever Γ' can be obtained by deleting some number of declarations from Γ. We sometimes abuse notation by writing Γ, Γ' for the concatenation of two contexts.

Our rules are formulated in the style of Pfenning [Pfe95], itself similar to **G3i** of [TS00]; the exact formulation of rules is immaterial to the applicability of our technique, although we feel that this presentation corresponds to an especially natural notion of proof term. We depart slightly from convention by denoting the occurrence of F in Γ using the premiss $h{:}F \in \Gamma$, rather than writing Γ as $\Gamma', h{:}A, \Gamma''$; we feel that this presentation makes the correspondence between proof-rules and proof terms more transparent. IID_0's core proof rules are below.

$$\frac{h{:}F \in \Gamma}{\Gamma \vdash F}\, axiom \qquad \frac{\Gamma \vdash F \quad \Gamma, h{:}F \vdash G}{\Gamma \vdash G}\, cut \qquad \frac{\Gamma \vdash F \quad \Gamma \vdash G}{\Gamma \vdash F \wedge G} \wedge r \qquad \frac{\Gamma \vdash F_i}{\Gamma \vdash F_1 \vee F_2} \vee r_i$$

$$\frac{\Gamma, h{:}F \vdash G}{\Gamma \vdash F \Rightarrow G} \Rightarrow r \qquad \frac{\Gamma, x \vdash F}{\Gamma \vdash \forall x.F} \forall r \qquad \frac{\Gamma \vdash F[t/x]}{\Gamma \vdash \exists x.F} \exists r \qquad \frac{\Gamma, h{:}F_i \vdash G \quad h'{:}F_1 \wedge F_2 \in \Gamma}{\Gamma \vdash G} \wedge l_i$$

$$\frac{\Gamma, h_1{:}F_1 \vdash G \quad \Gamma, h_2{:}F_2 \vdash G \quad h'{:}F_1 \vee F_2 \in \Gamma}{\Gamma \vdash G} \vee l \qquad \frac{\Gamma, F[t/x] \vdash G \quad h'{:}\forall x.F \in \Gamma}{\Gamma \vdash G} \forall l$$

$$\frac{\Gamma \vdash F_1 \quad \Gamma, h{:}F_2 \vdash G \quad h'{:}F_1 \Rightarrow F_2 \in \Gamma}{\Gamma \vdash G} \Rightarrow l \qquad \frac{\Gamma, x, h{:}F \vdash G \quad h'{:}\exists x.F \in \Gamma}{\Gamma \vdash G} \exists l$$

The $\forall l$ and $\exists r$ rules have a free variable side condition: every free variable of t is contained in Γ. Note that the eigenvariable side condition for $\forall r$ and $\exists l$ (i.e. $x \notin \Gamma$) is enforced by our notion of well-formed context; this notion gives rise to a similar side condition for the hypothetical variables introduced in $\Rightarrow r$ and in all of the left rules.

IID_0 is parameterized not only by its predicate symbols, but by the proof rules that refer to its atomic formulas. As is the case with compound formulas, the proof rules for atomic formulas can be divided into right rules and left rules. The right rules are supplied by the user, but must follow a particular format; the left rules are obtained algorithmically from the right rules.

Right rules. In general, the atomic right rules must be of the form:

$$\frac{\Gamma \vdash A_1 \quad \ldots \quad \Gamma \vdash A_n}{\Gamma \vdash A_0}$$

where $A_0 \dots A_n$ are built up from predicate symbols, term-schematic variables, and constructors from the term algebra. For example, the right rules for *nat*, *eq*, and $\bot$ are below (note that $\bot$ has no right rules).

$$\frac{}{\Gamma \vdash nat(z)}\, natr_z \qquad \frac{\Gamma \vdash nat(t)}{\Gamma \vdash nat(succ\ t)}\, natr_s \qquad \frac{}{\Gamma \vdash eq(t,t)}\, eqr$$

Left rules. In the setting of functional programming, a catamorphism (the most famous of which is probably `foldr`) can be used to provide a canonical notion of induction over the inhabitants of an inductively defined datatype. Through the lens of the Curry-Howard isomorphism, we can think of IID_0's predicate symbols as being inductively defined datatypes, the atomic right rules as datatype constructors, and the atomic left rules as catamorphisms (this analogy will be made more precise in 3.2). Just as `foldr` has arguments corresponding to the constructors `nil` and `cons`, so too must the atomic left rule for the predicate symbol *nat* have *minor premisses* corresponding to $natr_z$ and $natr_s$. In general, atomic left rules all have the form

$$\frac{\begin{array}{c}minor\\premisses\end{array} \quad \Gamma, h{:}P(t_1, \dots, t_n) \vdash G \quad h'{:}a(t_1, \dots, t_n) \in \Gamma}{\Gamma \vdash G}$$

where the *major premiss* (i.e. $\Gamma, h{:}P(t_1, \dots, t_n) \vdash G$) can be seen as allowing for the generalization of the induction hypothesis P, and the *minor premisses* are calculated from the right rules. For example, the atomic left rules for *nat*, $\bot$, and *eq* are defined as follows.

$$\frac{\Gamma \vdash P(z) \quad \Gamma, x, h_0{:}P(x) \vdash P(succ\ x) \quad \Gamma, h{:}P(t) \vdash G \quad h'{:}nat(t) \in \Gamma}{\Gamma \vdash G}\, natl$$

$$\frac{\Gamma, h{:}F \vdash G \quad h'{:}\bot \in \Gamma}{\Gamma \vdash G}\, \bot l \qquad \frac{\Gamma, x \vdash P(x,x) \quad \Gamma, h{:}P(t,u) \vdash G \quad h'{:}eq(t,u) \in \Gamma}{\Gamma \vdash G}\, eql$$

The minor premisses of *natl* are calculated from $natr_z$ and $natr_s$; the sole minor premiss of *eql* is calculated from *eqr*; $\bot$ has no right rules, and therefore $\bot l$ has no minor premisses. For details as to how minor premisses are calculated from right rules and how mutual dependencies are treated, we refer the reader to [ML71] and [Bro06]. We can obtain Leibniz equality

$$\frac{\Gamma, h{:}Q(t) \Rightarrow Q(u) \vdash G \quad h'{:}eq(t,u) \in \Gamma}{\Gamma \vdash G}$$

from *eql* by instantiating $P(x,y)$ to a predicate of the form $Q(x) \Rightarrow Q(y)$, and satisfying the minor premiss with the *impr* rule; we can obtain the more familiar version of $\bot l$

$$\frac{h{:}\bot \in \Gamma}{\Gamma \vdash G}$$

by instantiating F to G and satisfying the major premiss with the *axiom* rule. In fact, the major premiss of every atomic left rule can be made redundant this way, and presenting IID_0's atomic left rules without major premisses would not complicate the presentation of our consistency proof. However, as Brotherston points out in [Bro06], without such major premisses, the cut rule could not be eliminated from all of IID_0's proofs; although proving full cut elimination for IID_0 is beyond the scope of this paper, we choose this formulation to preserve the possibility for future work. IID_0 has the following basic properties:

Lemma 2.

(Weakening) If $\Gamma \vdash F$ *and* $\Gamma' \geq \Gamma$ *then* $\Gamma' \vdash F$
(Exchange) If $\Gamma, h{:}G, h'{:}G', \Gamma' \vdash F$ *then* $\Gamma, h'{:}G', h{:}G, \Gamma' \vdash F$
(Contraction) If $\Gamma_0, h{:}G, \Gamma_1, h'{:}G, \Gamma_2 \vdash F$ *then* $\Gamma_0, h{:}G, \Gamma_1, \Gamma_2 \vdash G$ *and* $\Gamma_0, \Gamma_1, h'{:}G, \Gamma_2 \vdash F$
(Substitution) If $\Gamma, x, \Gamma' \vdash F$ *and* $fv(t) \subseteq \Gamma$ *then* $\Gamma, \Gamma'[t/x] \vdash G[t/x]$

Definition 3 (Right Normal). *We say that a proof of* $\Gamma \vdash F$ *is* right-normal *iff it contains only right rules.*

Note right-normality is a stronger criterion than being cut-free. We are primarily concerned with right-normal proofs of atomic formulas; the following lemma summarizes the interaction between atomic right-normal proofs and minor premisses in the left rules of IID_0.

Lemma 3 (Folding Right-Normal Proofs). *Given a right-normal proof of* $\Gamma \vdash a(t_1, \ldots, t_n)$ *and proofs of the minor premisses of* a*'s left rule with induction hypothesis* P*, then* $\Gamma \vdash P(t_1, \ldots, t_n)$ *is provable using only substitution instances of the minor premisses' proofs and cut.*

Proof. By a structural induction on $\Gamma \vdash a(t_1, \ldots, t_n)$, using substitution.

It is easy to see that Lemma 3 generalizes to the mutual dependent case.

3.2 Proof Terms

We define proof terms for Martin Löf's IID_0 in order to provide a concise notation for describing and manipulating proof trees. The syntax for the non-atomic proof terms is defined below.

$$\begin{aligned} C, D, E ::= {} & \mathit{axiom}\ h \mid \mathit{cut}\ F\ C\ (h.D) \mid \mathit{andr}\ C\ D \mid \mathit{andl}_i\ (h.C)\ h' \mid \mathit{orr}_i\ C \mid \\ & \mathit{orl}\ (h_1.C)\ (h_2.D)\ h' \mid \mathit{impr}\ (h.C) \mid \mathit{impl}\ C\ (h.D)\ h' \mid \mathit{allr}\ (x.C) \mid \\ & \mathit{alll}\ t\ (h.C)\ h' \mid \mathit{existsr}\ t\ C \mid \mathit{existsl}\ (x.h.C)\ h' \mid \ldots \end{aligned}$$

We interpret $h.C$ and $x.C$ as binding occurrences of the variables h and x in C, and write $C[t/x]$ for the capture avoiding substitution of t for x. The typing

rules are given below.

$$\frac{h{:}F \in \Gamma}{\Gamma \vdash axiom\ h : F} \qquad \frac{\Gamma \vdash C : F \quad \Gamma, h{:}F \vdash D : G}{\Gamma \vdash cut\ F\ C\ (h.D) : G} \qquad \frac{\Gamma \vdash C : F \quad \Gamma \vdash D : G}{\Gamma \vdash andr\ C\ D : F \wedge G}$$

$$\frac{\Gamma \vdash C : F_i}{\Gamma \vdash orr_i\ C : F_1 \vee F_2} \qquad \frac{\Gamma, h{:}F \vdash C : G}{\Gamma \vdash impr\ (h.C) : F \Rightarrow G} \qquad \frac{\Gamma, x \vdash C : F}{\Gamma \vdash allr\ (x.C) : \forall x.F}$$

$$\frac{\Gamma \vdash C : F[t/x]}{\Gamma \vdash existsr\ t\ C : \exists x.F} \qquad \frac{\Gamma, h{:}F[t/x] \vdash G \quad h'{:}\forall x.F \in \Gamma}{\Gamma \vdash alll\ t\ (h.C)\ h' : G} \qquad \frac{\Gamma, x, h{:}F \vdash G \quad h'{:}\exists x.F \in \Gamma}{\Gamma \vdash existsl\ (x.h.C)\ h' : G}$$

$$\frac{\Gamma, h{:}F_i \vdash C : G \quad h'{:}F_1 \wedge F_2 \in \Gamma}{\Gamma \vdash andl_i\ (h.C)\ h' : G} \qquad \frac{\Gamma \vdash C : F_1 \quad \Gamma, h{:}F_2 \vdash D : G \quad h'{:}F_1 \Rightarrow F_2 \in \Gamma}{\Gamma \vdash impl\ C\ (h.D)\ h' : G}$$

$$\frac{\Gamma, h_1{:}F_1 \vdash C : G \quad \Gamma, h_2{:}F_2 \vdash D : G \quad h'{:}F_1 \vee F_2 \in \Gamma}{\Gamma \vdash orl\ (h_1.C)\ (h_2.D)\ h' : G}$$

The proof term constructors corresponding to atomic inference rules are obtained in a manner analogous to the above: subtrees correspond to subterms, and extending the context of a subderivation corresponds to introducing a new bound variable; term-, formula-, and predicate-schematic variables are explicitly included in proof terms when convenient. We continue our example below.

$$C, D, E ::= \ldots \mid natr_z \mid natr_s\ t\ C \mid eqr\ t \mid natl\ P\ t\ C_0\ (x.h_0.C_1)\ (h.D)\ h' \mid \\ botl\ F\ (h.C)\ h' \mid eql\ P\ t\ u\ (x.C)\ (h.D)\ h'$$

Due to space constraints, we omit the typing rules for atomic proof terms; it should be a straightforward exercise for the reader to reconstruct them.

Theorem 2 (Curry-Howard). *There exists an isomorphism between derivations of $\Gamma \vdash F$ and proof terms D s.t. $\Gamma \vdash D : F$.*

Proof. Each direction is proven by a straightforward induction on the given derivation, replacing each instance of a proof rule with the corresponding typing rule, and vice versa.

The Curry-Howard isomorphism justifies treating well-typed proof terms and proof derivations interchangeably, without losing generality. For example, Lemma 3 corresponds to an analogous lemma over well-typed terms, whose behavior on *untyped* proof terms is, in the case of *nat*, realized by the following function.

```
fold-nat (natr_z) (P) (D_1) (x.h.D_2) = D_0
fold-nat (natr_s t C) (P) (D_1) (x.h.D_2) =
  cut P(t) (fold-nat C P D_1 (x.h.D_2)) h.(D_2[t/x])
```

4 The Consistency Proof

We give now the consistency proof of Martin-Löf's IID_0, where we reason *about* derivations using lexicographic path induction. Informally, a logic is consistent if its notion of provability is somehow nontrivial. Although there are several ways to make this precise, one of the more common is to demonstrate the existence of an unprovable sequent, which can typically be shown as a corollary to some sort of normalization theorem. We generalize this notion slightly by proving that, if $\cdot \vdash C : A$ then there exists a right-normal D s.t. $\cdot \vdash D : A$. If, for example, $\bot$ is an atomic predicate symbol, then consistency in the traditional sense follows immediately.

Although proof terms are in a sense finite trees, we do not apply lexicographic path induction to them directly because proof terms contain information that we do not consider relevant to the size of a proof. Instead, we apply the principle to *skeletons* of proof trees, which will be defined in 4.1; lexicographic path induction on skeletons subsumes structural induction on proof trees.

In many ways, our proof follows the same general structure as Gentzen's proof of the consistency of arithmetic [Gen38], and the Howard [How70] and Schütte [Sch77] proofs of normalization for Gödel's T. All involve assigning well-founded orderings to proof trees/λ-calculus terms (ϵ_0 for Gentzen, Howard and Schütte, the LPO here); all demonstrate normalization for a restricted class of sequents/types ($\cdot \vdash \cdot$ for Gentzen; the base type for Howard and Schütte, $\cdot \vdash A$ here); and all unfold inductions all-at-once, rather than one-at-a-time. Our proof differs from the others in that lexicographic path induction is applied directly to skeletons of proof terms, whereas in Gentzen's, Howard's and Schütte's proofs, the assignment of ordinals to proofs/λ-calculus terms is very complex. This discrepancy shouldn't be too surprising: the order type of the lexicographic path ordering approaches the small Veblen ordinal [DO88,Mos04], which is ***MUCH*** larger than ϵ_0; it is often the case that using stronger-than-necessary assumptions can lead to simpler proofs.

4.1 Ordering Proof Terms

Although we have seen in 3.2 that well-typed proof terms contain the same amount of information as proof trees, we do not consider all of this information to be relevant to the size of proofs. In particular, we do not consider hypotheses or elements of the term algebra relevant, nor, with the notable exception of *cut*, do we consider formulas or predicates relevant. We therefore map proof terms into labeled trees, called *skeletons*, which are obtained from proof terms by *stripping* spurious information. Because we consider the size of cut-formulas to be relevant to the size of proofs (as is typically the case), skeletons are defined for formulas as well. The signature Σ for skeletons is defined as follows.

$$\begin{aligned}
\Sigma^0 &= \mathsf{a}, \mathsf{axiom} \\
\Sigma^1 &= \mathsf{all}, \mathsf{exists}, \mathsf{andl}_i, \mathsf{orr}_i, \mathsf{impr}, \mathsf{allr}, \mathsf{alll}, \mathsf{existsr}, \mathsf{existsl} \\
\Sigma^2 &= \mathsf{and}, \mathsf{or}, \mathsf{imp}, \mathsf{andr}, \mathsf{orl}, \mathsf{impl} \\
\Sigma^3 &= \mathsf{cut}
\end{aligned}$$

Σ also contains constructors of the appropriate arity corresponding to atomic rules, where the stripping function is defined analogously. We give some representative cases of stripping. Note the use of the sans serif font for skeletons.

$$
\begin{aligned}
|a(t_1, \ldots, t_n)| &= \mathsf{a} \\
|F \wedge G| &= \mathsf{and}(|F|, |G|) \\
|\forall x.F| &= \mathsf{all}(|F|) \\
|cut\ F\ C\ h.D| &= \mathsf{cut}(|F|, |C|, |D|) \\
|axiom\ h| &= \mathsf{axiom} \\
|allr\ x.C| &= \mathsf{allr}(|C|) \\
|existsl(x.h.C)\ t| &= \mathsf{existsl}(|C|) \\
|natl\ P\ t\ C_0\ (x.h.C_1)\ (h.D)\ h'| &= \mathsf{natl}(|C_0|, |C_1|, |D|)
\end{aligned}
$$

Definition 4 (Skeleton Ordering). *We define* $<$ *as the least* transitive *ordering on* Σ *satisfying all of the following:*

1. *If* f *corresponds to an atomic formula, and* g *is any of* {and, or, imp, all, exists} *then* $\mathsf{f} < \mathsf{g}$
2. *If* f *corresponds to a formula, and* g *corresponds to a proof term, then* $f < g$
3. *If* f *corresponds to a right-rule or a compound left-rule then* $\mathsf{f} < \mathsf{cut}$
4. *If* f *corresponds to an atomic left-rule, then* $\mathsf{cut} < \mathsf{f}$
5. *If* f *corresponds to an atomic right-rule, then* $\mathsf{f} < \mathsf{axiom}$

$<_{\mathsf{lpo}}$ *is the lifting of* $<$ *to skeletons via the LPO.*

The first two clauses of Definition 4 are motivated by the first two clauses of Lemma 4, which will be used by Lemma 6; the third clause is motivated by Lemma 6 as well. The last two clauses are motivated by Lemma 5.

Lemma 4 (Properties of Skeletons).

1. *For every* A *and every* K, $|A| <_{\mathsf{lpo}} |K|$
2. *For every* F *and every* C, $|F| <_{\mathsf{lpo}} |C|$
3. *For every* C, *every* t *and every* x, $|C| = |C[t/x]|$

Proof. Each case is a straightforward by structural induction; both 1 and 2 are instances of the big head principle.

Because the LPO has the subterm property, and because skeletons contain most of the structural information of proof terms, all of the instances of structural induction on C in this paper can be replaced by lexicographic path induction on $|C|$.

4.2 The Normalization Procedure

Our proof is structured as follows. Given $\cdot \vdash C : A$, C must either be of the form $ar\ C_1 \dots C_n$, or $cut\ F\ D\ h.E$. In the former case, we right-normalize $C_1 \dots C_n$, and apply ar to the result. In the latter case, we find a proof term C' which is smaller than C and right-normalize C'; however, the calculation of C' depends on whether the cut-formula F is atomic or compound. Observe that, if F is atomic, then, by induction, we can right-normalize D into D'; C' is obtained by eliminating all uses of h from $h.E$, making use of D' and Lemma 3. If F is compound, we perform what is essentially a small-step version of the cut-admissibility proof in [Pfe95], where, for reasons that will be explained later, we must be careful to avoid any "commutative conversions" for atomic left rules.

Recall that we write A for atomic formulas and K for for compound formulas.

Lemma 5 (Atomic Cut Reduction). *For every C and $h.D$, if C is right-normal, $\Gamma \vdash C : A$ and $\Gamma, h{:}A \vdash D : F$, then there exists E such that $\Gamma \vdash E : F$ and $|E| <_{\mathsf{lpo}} |D|$.*

Proof. By structural induction on D. Most cases are straightforward, often using weakening on C and exchange on D before applying the induction hypothesis, and using monotonicity of $<_{\mathsf{lpo}}$ on the result. The non-trivial uses of h come from *axiom* and the atomic left rule for A. If $D = axiom\ h$, we return C, which is smaller than $axiom\ h$ by the big head principle and the fifth clause of Definition 4. If D is a non-trivial instance of an atomic left rule, we induct D's subterms and apply Lemma 3, whose output is smaller than D by the third clause of Lemma 4, the big head principle and the fourth clause of Definition 4. The proof can be realized on untyped terms by the function `redA`, some representative cases of which are below.

```
redA (C) (h.axiom h) = C
redA (C) (h.axiom h') = axiom h'
redA (C) (h.natl P t D0 (x.h'.D1) (h''.D2) h) =
  let D0' = redA C h.D0
      (x.h'.D1') = x.h'.(redA C h.D1)
      E = fold-nat C P D0' (x.h'.D1')
      (h''.D2') = h''.(redA C h.D2)
  in
    cut P(t) E (h''.D2')
redA (C) (h.natl P t D0 (x.h'.D1) (h''.D2) h''') =
  let D0' = redA C h.D0
      (h'.D1') = x.h'.(redA C h.D1)
      (h''.D2') = h''.(redA C h.D2)
  in
   natl P t D0' (x.h'.D1') (h''.D2') h'''
```

Lemma 6 (Compound Cut Reduction). *For every Γ which only contains compound assumptions (i.e. if $h{:}F \in \Gamma$, then F must be compound), if $\Gamma \vdash cut\ K\ C\ (h.D) : G$ then there exists E such that $\Gamma \vdash E : G$ and $|E| <_{\mathsf{lpo}} |cut\ K\ C\ (h.D)|$.*

Proof. By structural induction on $cut\ K\ C\ (h.D)$. The proof can be realized on untyped terms by the function `redK`, some representative cases of which are below. Most cases use Lemma 4, clause 2.

If C is a left rule, or if D is either right rule or left rule that acts on a hypothesis other than h, then the *cut* is "commutative," and the offending rule will be bubbled up (see Definition 4, clause 3). Note that the restriction on Γ means that we never encounter commutative cuts of atomic left rules, and thus will never have to bubble one past a *cut*. This is critical, because as we have seen in Lemma 5, `cut` must be smaller than `al`.

$$\texttt{redK}\ (cut\ K\ C\ (h.andr\ D_1\ D_2)) = andr\ (cut\ K\ C\ h.D_1)\ (cut\ K\ C\ h.D_2)$$
$$\texttt{redK}\ (cut\ K\ (andl_i\ (h'.C)\ h'')\ (h.D)) = andl_i\ (h'.cut\ K\ C\ h.D)\ h''$$
$$\texttt{redK}\ (cut\ K\ C\ (h.andl_i\ (h'.D)\ h'')) = andl_i\ (h'.cut\ K\ C\ h.D)\ h''$$

If C is a right rule, and D is a left rule that acts on h, then the cut is "essential." The sizes of cut-formulas play a crucial role in these cases. The $\forall$ and $\exists$ essential cases use Lemma 4, clause 3.

$$\begin{aligned}&\texttt{redK}\ (cut\ (F \Rightarrow G)\ (impr\ h_0.C_0)\ (h.impl\ D_0\ (h_1.D_1)\ h)) =\\ &\quad cut\ G\ (cut\ F\ (cut\ (F \Rightarrow G)\ (impr\ h_0.C_0)\ h.D_0)\ C_0)\\ &\qquad\quad (h_1.cut\ (F \Rightarrow G)\ (impr\ h_0.C_0)\ h.D_1)\\ &\texttt{redK}\ (cut\ (\forall x.F)\ (allr\ x.C)\ (h.alll\ t\ (h'.D)\ h)) =\\ &\quad cut\ (F[t/x])\ (C[t/x])\ (h'.cut\ (\forall x.F)\ (allr\ x.C)\ h.D)\end{aligned}$$

If C or D is a compound cut, we apply the induction hypothesis and monotonicity of $<_{\mathsf{lpo}}$; if C or D is an atomic cut, we bubble it up (see Lemma 4, clause 1).

$$\texttt{redK}\ (cut\ K\ (cut\ L\ C_0\ h'.C_1)\ D) = cut\ K\ \texttt{(redK}\ C)\ h.D$$
$$\texttt{redK}\ (cut\ K\ C\ (h.cut\ L\ D_0\ h'.D_1)) = cut\ K\ C\ h.\texttt{(redk}\ D)$$
$$\texttt{redK}\ (cut\ K\ h.(cut\ A\ C_0\ h'.C_1)\ D) = cut\ A\ C_0\ (h'.cut\ K\ C_1\ h.D)$$
$$\begin{aligned}&\texttt{redK}\ (cut\ K\ C\ (h.cut\ A\ D_0\ h'.D_1)) =\\ &\quad cut\ A\ (cut\ K\ C\ h.D_0)\ h'.(cut\ K\ C\ h.D_1)\end{aligned}$$

We are now ready to prove our consistency theorem.

Theorem 3 (Consistency of $\mathbf{IID}_0$). *For all C, if $\cdot \vdash C : A$ then there exists a right-normal D such that $\cdot \vdash D : A$*

Proof. By lexicographic path induction on $|C|$. This theorem can be realized by the function `norm` on untyped terms, which is defined as follows.

$$\texttt{norm}\ (ar\ C_1\ \ldots\ C_n) = ar\ (norm\ C_1)\ \ldots\ (norm\ C_n)$$
$$\texttt{norm}\ (cut\ A\ C_1\ h.C_2) = \texttt{norm (redA (norm}\ C_1)\ (h.C_2))$$
$$\texttt{norm}\ (C\ \texttt{as}\ cut\ K\ C_1\ h.C_2) = \texttt{norm (redK}\ C)$$

Note that the consistency theorem can be generalized straightforwardly to sequents of the form $\Gamma \vdash F$, where Γ contains only eigenvariables, and F contains no implications; we choose not to formulate it this way for the sake of simplicity. Also note that, if we choose to omit the atomic left rule for a predicate symbol a, then the consistency theorem can also be generalized to sequents of the form $\Gamma \vdash A$, where Γ contains only eigenvariables and hypotheses of the form $a(t_1, \ldots, t_n)$; this generalization is used in the proof of weak normalization for Gödel's T.

4.3 Heyting Arithmetic

Heyting arithmetic is usually presented with a term algebra that contains constructors not only for z and *succ*, but also for addition and multiplication, whose operational meaning is defined axiomatically; induction is usually defined as an axiom schema of the form $P(z) \wedge (\forall x.P(x) \Rightarrow P(succ\ x)) \Rightarrow \forall x.P(x)$. We instead formulate Heyting Arithmetic as an instance of IID_0, whose term algebra includes only z and *succ*, and whose atomic formulas include nat, eq and $\bot$ (as described in 3.1) along with predicate symbols for $add, mult$ and any other functions that we wish to reason about. The usual axioms for reasoning about addition and multiplication are instead formulated as right rules for *add* and *mult*.

$$\frac{}{\Gamma \vdash add(z,t,t)}\, addr_z \qquad \frac{\Gamma \vdash add(t_1,t_2,t_3)}{\Gamma \vdash add(succ\ t_1, t_2, succ\ t_3)}\, addr_s$$

$$\frac{}{\Gamma \vdash mult(z,t,z)}\, multr_z \qquad \frac{\Gamma \vdash mult(t_1,t_2,t_3) \quad \Gamma \vdash add(t_1,t_3,t_4)}{\Gamma \vdash mult(succ\ t_1, t_2, t_4)}\, multr_s$$

Instead of performing induction on terms, we *relativize* quantifiers (as in [Bro06]), and perform induction on *nat* using *natl*. For example, the relativization of $\forall x_1.\forall x_2.\exists y.add(x_1,x_2,y)$ is $\forall x_1.nat(x_1) \Rightarrow \forall x_2.nat(x_2) \Rightarrow \exists y.nat(y) \wedge add(x_1,x_2,y)$, which is a theorem in this logic.

Unfortunately, two of Peano's axioms (the inequality of zero and one, and the injectivity of the successor operation) are not provable in the logic as we have described it thus far. Thus, we explicitly add the following inference rules, *without* changing any of the existing atomic left rules.

$$\frac{\Gamma \vdash eq(t, succ\ t)}{\Gamma \vdash \bot}\, pa7 \qquad \frac{\Gamma \vdash eq(succ\ t, succ\ u)}{\Gamma \vdash eq(t,u)}\, pa8$$

Fortunately, the addition of these two rules barely complicates the consistency proof: `redA` must be extend with two trivial inductions over $pa7$ and $pa8$, and the normalization algorithm must be extended as follows:

```
norm (pa7 C) =  % undefined: no such right-normal C
norm (pa8 C) = let (eqr (succ t)) = norm C in eqr t
```

4.4 Weak Normalization for Gödel's T

Gödel's T is the extension of the simply-typed λ-calculus with terms for zero, successor, and a primitive recursion operator. Due to space constraints, we cannot include a full development of the proof of weak normalization for Gödel's T, but the proof follows a structural relations argument that is nearly identical to that of the simply-typed λ-calculus found in [SS08]. The proofs of Closure Under Weak Head Expansion and the Escape Lemma, both of which proceed by induction on types, are entirely unchanged. The major challenge is in the Fundamental Theorem, where we must show that applications of the primitive

recursion operator are in the logical relation. The primitive recursion operator defines induction on terms of base type, so it should come as no surprise that the Fundamental Theorem can be proven for this case by appealing to induction on the notion of "weakly normalizing at base type," which is represented here by the unary predicate symbol hc^o. To this end, we extend the assertion logic with an IID_0-style left rule for hc^o (and only hc^o), which allows us to complete the proof of the Fundamental Theorem. A slight generalization of Theorem 3 can be applied to this assertion logic, thus completing the proof of weak normalization. For details, see `http://www.twelf.org/lpo/`.

5 Conclusion

We have demonstrated that lexicographic path induction is a suitable replacement for transfinite induction in at least some settings. The proofs of consistency for Heyting arithmetic and Gödel's T have been formalized in a prototypical extension of Twelf.

We are not the first to prove the consistency of a logic using an ordering from term rewriting theory: [DP98,Bit99,Urb01] show the strong normalization of cut elimination for different formulations of first-order logic using either the recursive (a.k.a multiset) path ordering or lexicographic path ordering. However, because these results rely on proving termination of term rewriting systems, they cannot be scaled to arithmetic (by Gödel's second incompleteness theorem and [Buc95]).

Several other logics from the programming languages literature formulate the notion of induction similar to IID_0: the proofs of consistency for $\mathrm{FO}\lambda\Delta^{\mathbb{N}}$ [MM00], Linc [MT03], and $\mathcal{G}$ [GMN08] all rely on logical relations; the proof of consistency for LKID [Bro06] uses model theory. We are optimistic that at least some of these systems can be proven consistent using lexicographic path induction.

The proof theoretic strength of $\mathrm{IID}_{<\omega}$([ML71], section 10) exceeds that of the small Veblen ordinal, and thus its consistency cannot be proven by lexicographic path induction. We leave whether our technique scales to any useful logics or type theories whose proof-theoretic ordinal is greater than ϵ_0, but smaller than small Veblen, to future work.

Acknowledgments: We would like to thank Michael Rathjen and Georg Moser for their helpful answers to our questions regarding large ordinals, and Søren Debois for his helpful comments on an earlier draft of this paper.

References

[Bit99] Elias Tahhan Bittar. Strong normalization proofs for cut-elimination in gentzens sequent calculi. In *Banach Center Publication*, pages 179–225, 1999.

[Bro06] James Brotherston. *Sequent Calculus Proof Systems for Inductive Definitions.* PhD thesis, University of Edinburgh, November 2006.

[Buc95] Wilfried Buchholz. Proof-theoretic analysis of termination proofs. *Ann. Pure Appl. Logic*, 75(1-2):57–65, 1995.

[DO88] Nachum Dershowitz and Mitsuhiro Okada. Proof-theoretic techniques for term rewriting theory. In *LICS*, pages 104–111. IEEE Computer Society, 1988.

[DP98] Roy Dyckhoff and Luis Pinto. Cut-elimination and a permutation-free sequent calculus for intuitionistic logic. *Studia Logica*, 60(1):107–118, 1998.

[Dyb91] Peter Dybjer. Inductive sets and families in martin-lof's type theory and their set-theoretic semantics. In *Logical Frameworks*, pages 280–306. Cambridge University Press, 1991.

[Gen38] Gerhard Gentzen. New version of the consistency proof for elementary number theory. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen, 1969*, pages 252–286. North-Holland Publishing Co., Amsterdam, 1938.

[GMN08] Andrew Gacek, Dale Miller, and Gopalan Nadathur. Combining generic judgments with recursive definitions. In F. Pfenning, editor, *Proceedings of LICS 2008*, pages 33–44. IEEE Computer Society Press, June 2008.

[How70] W. A. Howard. Assignment of ordinals to terms for primitive recursive functions of finite type. In A.Kino, J. Myhill, and R. E. Vesley, editors, *Intuitionism and Proof Theory*, pages 443–458. North-Holland, 1970.

[KL80] Sam Kamin and Jean-Jacques Levy. Attemps for generalising the recursive path orderings. Unpublished lecture notes, 1980.

[ML71] Per Martin-Löf. Hauptsatz for the intuitionistic theory of iterated inductive definitions. In J.E. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*. North Holland, 1971.

[MM00] Raymond McDowell and Dale Miller. Cut-elimination for a logic with definitions and induction. *Theoretical Computer Science*, 232(1-2):91–119, 2000.

[Mos04] Ingo Lepper Georg Moser. Why ordinals are good for you. In *ESSLLI 2003 - Course Material II*, volume 6 of *Collegium Logicum*, pages 1–65. The Kurt Gödel Society, 2004.

[MT03] Alberto Momigliano and Alwen Tiu. Induction and co-induction in sequent calculus. In Stefano Berardi, Mario Coppo, and Ferruccio Damiani, editors, *TYPES*, volume 3085 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2003.

[Pfe95] Frank Pfenning. Structural cut elimination. In D. Kozen, editor, *Proceedings of the Tenth Annual Symposium on Logic in Computer Science*, pages 156–166, San Diego, California, June 1995. IEEE Computer Society Press.

[Pie02] Benjamin C. Pierce. *Types and programming languages*. MIT Press, Cambridge, MA, USA, 2002.

[PM93] C. Paulin-Mohring. Inductive Definitions in the System Coq - Rules and Properties. In M. Bezem and J.-F. Groote, editors, *Proceedings of the conference Typed Lambda Calculi and Applications*, number 664 in Lecture Notes in Computer Science, 1993. LIP research report 92-49.

[Sch77] K. Schütte. *Proof Theory*. Springer-Verlag, 1977.

[SS08] Carsten Schürmann and Jeffrey Sarnat. Structural logical relations. In *LICS*, pages 69–80, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[TS00] A.S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Number 43 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, second edition edition, 2000.

[Urb01] Christian Urban. Strong normalisation for a gentzen-like cut-elimination procedure. In *Proceedings of the 5th International Conference on Typed Lambda Calculi and Applications*, pages 415–42, May 2001.