Ground Stratified Inductive Definitions

Nathan Guermond
University of Minnesota Twin-Cities, USA

Gopalan Nadathur
University of Minnesota Twin-Cities, USA

Abstract

Logics of definitions extend first order intuitionistic logic with fixed-point definitions which associate formulas to atomic predicates. These associated formulas must be constrained for consistency. In the original formulation, predicate symbols were required to be ordered and only predicates lower in the order were allowed to appear negatively in the defining formula. This constraint renders ineligible definitions such as those of logical relations in which the predicate being defined must be allowed to appear negatively, albeit with smaller arguments. Tiu has formalized a weaker constraint called ground stratification that permits such definitions and has shown that it suffices for consistency. Definitions can also be given a least fixed-point interpretation via a special induction rule. We address the question of whether Tiu's relaxation carries over to such a treatment. We propose a new induction rule for ground stratified inductive definitions that takes into account the fact that the definition of the predicate in question must itself be considered to be stratified by the complexity of its arguments to obtain a least fixed-point interpretation. We establish the consistency of the resulting logic and we illustrate its new capabilities via an example that encodes a strong normalizability proof for the simply typed λ-calculus in which the reducibility predicate is inductively defined by recursion on its arguments.

2012 ACM Subject Classification Theory of computation → Proof theory; Theory of computation → Logic and verification

Keywords and phrases Logic of definitions, induction, cut-elimination, consistency

Digital Object Identifier 10.4230/LIPIcs.FSCD.2026.18

Supplementary Material *Software (Source Code)*: `https://github.com/Sparrow-Project/strong-normalization`, archived at `swh:1:dir:c0f840814a7eded82dfb5f1d4d8a27af3c9bc868`

1 Introduction

Logics of definitions, originally introduced by McDowell and Miller [14], extend conventional predicate logics with the capability of treating predicate constants as defined symbols. Formally, this treatment is realized by associating formulas with predicate constants through a collection of clauses and including rules for using these clauses to "introduce" atomic formulas in the course of derivations. In the simplest form, the rules build in a fixed-point treatment of the definitions of predicates that are provided in this manner. However, as shown by [17], the inference rules can be refined to yield a least or greatest fixed-point interpretation to the definitions, thereby yielding capabilities for inductive and co-inductive reasoning in the logic. The resulting logics are well-suited to encoding reasoning about rule-based relational specifications, as has been amply established by experience with the proof assistant Abella [2] that implements a particular variant called $\mathcal{G}$ [7].

The definitions that are associated with predicate constants in the logics under consideration must be constrained in order to ensure their consistency. Specifically, a definition must not be circular in that it assumes its own existence in its construction. This requirement translates concretely into restrictions on the negative occurrences of the predicate constant in the formula defining it. The condition imposed in [14] is that there must be an ordering on predicate constants that determines which of them have been already defined and can

© Nathan Guermond and Gopalan Nadathur;
licensed under Creative Commons License CC-BY 4.0

therefore be used negatively in the definition of another predicate constant. Unfortunately, this strict stratification restriction based on predicate names is too limiting for some applications. A prime example of this is that of logical relations that are often used in reasoning about properties of programming languages. The definition of these relations are typically indexed by types, but may assume that the relation has already been defined for structurally smaller types.

In past work [11, 18], it has been shown that the restriction on fixed-point definitions can be weakened to a form that allows for building in of the arguments of an atomic formula into the stratification ordering while maintaining the stricter form of stratification for inductive definitions. The weaker condition has been called *ground stratification*. While this relaxation suffices for many applications, there are still some situations in which a ground stratified definition is required to be inductive.[1] Whether such a capability can be coherently supported is the issue we take up here. Our results in this context are threefold. First, we show that we can sensibly provide a least fixed-point reading to a ground stratified fixed-point definition but that, to do so, we must think in terms of *ground* (variable-free) instances of the defining clauses. Second, we show that the logic can be endowed with inference rules for these definitions that approximate their understanding in this manner while retaining consistency. An important aspect of our additions is that they preserve the earlier treatment of inductive definitions under the stricter stratification condition. Finally, we demonstrate that there is practical utility to the augmentation to the logic that results from our work.

The rest of the paper is organized as follows. The next section introduces the logic of fixed-point definitions. Section 3 discusses induction within this setting, culminating with inference rules for treating ground-stratified inductive definitions. The next two sections establish the consistency of the resulting logic and illustrate the usefulness of its new capabilities in formalizing a particular proof of strong normalizability for the simply typed lambda calculus. We conclude the paper with a discussion of related and future work. The paper includes two appendices, one of which elaborates on a discussion of least-fixed points associated with ground stratified inductive definitions and the other of which illustrates the use of ground-stratified inductive definitions in a proof of strong normalizability for System T. Formalizations of the examples in the paper within the Abella proof assistant are included as supplementary material.

2 A Logic of Fixed-Point Definitions

The logic of interest in this paper, which we call LD^{μ^+}, is an intuitionistic, first-order version of Church's Simple Theory of Types [4] that is extended by allowing atomic predicates to be interpreted via fixed-point or inductive definitions. We present the logic without inductive definitions below, delaying the introduction of induction to the next section. The logic that we present here is in fact identical to Tiu's logic LD without natural number induction [18]; the missing feature will be subsumed by the notion of induction that we introduce later.

2.1 Terms, Formulas, and Sequents

The syntax of the logic is based on that of Church's Simple Theory of Types. We assume a finite set of base types $\iota_1, \ldots, \iota_n$, function types $\alpha \to \beta$, and a distinguished type $\mathbf{o}$ of propositions. We further assume we are given a *signature* Σ, which is a set of type-annotated

[1] A specific example of this kind may be seen in the logical relation that is used in the proof of strong normalizability for System T [9]; see the remark at the end of Section 7.2 in [8].

constants of the form $\kappa : \tau$. The terms of the logic are then the well-typed terms in the simply typed lambda calculus that are constructed using the constants in the signature Σ and variables from a *variable context* $\mathcal{X}$, which is a finite set of distinct type-annotated variables $x : \tau$. Since a term t only exists with respect to a signature Σ and a variable context $\mathcal{X}$, and since Σ is fixed, we will say that t is well formed with respect to $\mathcal{X}$, or that t *lies over* $\mathcal{X}$. In particular this means that $\mathcal{X}$ must contain all free variables in t, but may contain more. A term is *ground* if it lies over the empty variable context $\emptyset$. We denote the set of ground terms of type α by ground(α). Assuming t lies over $\mathcal{X}$, we write $\mathcal{X} \vdash_\Sigma t : \tau$ to denote the fact that t has the type τ relative to $\mathcal{X}$ and Σ; we observe that there is a unique τ for which this relation holds. Terms are considered to be equal modulo α-, β-, and η-conversion. In what follows, we shall represent them by their normal forms modulo these rules, which are known to exist and to be unique up to α-conversion.

We call a type a *first-order* type if it does not contain $\mathbf{o}$, and a *predicate* type if it is $\mathbf{o}$ or of the form $\tau \to \omega$ for a first order type τ and a predicate type ω. We shall use ω, possibly with subscripts, to denote predicate types. If $p : \omega$ belongs to Σ, then p is said to be a predicate constant or symbol. If ω is $\mathbf{o}$, then p may also be referred to as a propositional symbol. A *formula* is a term of type $\mathbf{o}$. We assume that Σ contains the *logical constants* $\bot$ and $\top$ of type $\mathbf{o}$, $\wedge$, $\vee$, and $\supset$ of type $\mathbf{o} \to \mathbf{o} \to \mathbf{o}$ and, for every first-order type α, $\forall_\alpha$ and $\exists_\alpha$ of type $(\alpha \to \mathbf{o}) \to \mathbf{o}$. *Atomic* formulas are formulas of the form $p\ \vec{t}$ for some predicate symbol $p : \omega$ in Σ; here and elsewhere, we write $u\ \vec{t}$ as an abbreviation for the application $(\ldots(u\ t_1)\ \ldots\ t_n)$. We abbreviate $Q_\alpha(\lambda x.C)$, where Q_α is $\forall_\alpha$ or $\exists_\alpha$, by $Q_\alpha x.C$. We also follow the usual convention of writing $\wedge$, $\vee$, and $\supset$ in infix form in formulas in which they appear as the top-level logical symbol.

Given variable contexts $\mathcal{X}$ and $\mathcal{Y}$, a *substitution* $\theta = [t_1/x_1, \ldots, t_n/x_n]$ of type $\mathcal{Y} \to \mathcal{X}$ is an assignment of terms $t_1 : \tau_1, \ldots, t_n : \tau_n$ lying over $\mathcal{Y}$ to each variable $x_1 : \tau_1, \ldots, x_n : \tau_n = \mathcal{X}$. Such a substitution is said to be *for* $\mathcal{X}$, and its *range*, denoted by range(θ), is the smallest variable context containing all the free variables in $t_1, \ldots, t_n$. Moreover, it is said to be an $\mathcal{X}$*-grounding substitution* if $\mathcal{Y}$ is $\emptyset$. If t is a term lying over $\mathcal{X}$, the *application of* θ *to* t, written as $t\theta$, is the term $(\lambda x_1 \ldots \lambda x_n.t)\ \vec{t}$; this is evidently a term that lies over $\mathcal{Y}$. Given a variable context $\mathcal{X}$, we write $[t/x]_\mathcal{X}$ to denote the simultaneous substitution $[x_1/x_1, \ldots, x_n/x_n, t/x] : \mathcal{X} \to \mathcal{X}, x$, and we denote the trivial substitution by $\epsilon_\mathcal{X} : \mathcal{X} \to \mathcal{X}$.

Derivability for the logic is formalized in the sequent style. A sequent in this context is a judgment of the form $\mathcal{X} : \Gamma \vdash C$ for a finite multiset of formulas Γ and a formula C all of which lie over a finite variable context $\mathcal{X}$. We will assume all the types in $\mathcal{X}$ are first order. We will omit the type annotations in $\mathcal{X}$ when their specific identity is orthogonal to the discussion. The formulas in Γ are called the *context* or *assumption set* of the sequent, C is called its *conclusion* and the variables in $\mathcal{X}$ constitute its *eigenvariables*.

2.2 The Structural and Logical Rules

The rules for deriving sequents are of three kinds: those that explicate the meaning of the logical symbols, those that formalize the structural aspects of sequents, and those that interpret atomic predicates through their definitions. We discuss the first two kinds of rules in this subsection, leaving the elaboration of the last kind to the next subsection.

The rules corresponding to the logical symbols are identical in content to the ones to be found in usual first-order logics. We assume familiarity with the ones that build in the meanings of the logical connectives. The quantifier rules are shown in Figure 1. Γ denotes a multiset of formulas in these rules, and we write Γ, F to denote a multiset that comprises F and the formulas in Γ. Note also that, in the rules in Figure 1, we assume the variable x bound by the quantifier to be distinct from all the variables in $\mathcal{X}$, a requirement that can always be established by renaming the bound variable.

$$\frac{\mathcal{X} : \Gamma, C[t/x]_{\mathcal{X}} \vdash D \qquad \mathcal{X} \vdash_\Sigma t : \tau}{\mathcal{X} : \Gamma, \forall_\tau x.C \vdash D}\ \forall\mathcal{L} \qquad \frac{\mathcal{X}, x : \Gamma \vdash C}{\mathcal{X} : \Gamma \vdash \forall_\tau x.C}\ \forall\mathcal{R}$$

$$\frac{\mathcal{X}, x : \Gamma, C \vdash D}{\mathcal{X} : \Gamma, \exists_\tau x.C \vdash D}\ \exists\mathcal{L} \qquad \frac{\mathcal{X} : \Gamma \vdash C[t/x]_{\mathcal{X}} \qquad \mathcal{X} \vdash_\Sigma t : \tau}{\mathcal{X} : \Gamma \vdash \exists_\tau x.C}\ \exists\mathcal{R}$$

Figure 1 Logical rules for quantifiers.

$$\frac{\mathcal{X} : \Delta_1 \vdash B_1 \qquad \ldots \qquad \mathcal{X} : \Delta_n \vdash B_n \qquad \mathcal{X} : \Gamma, B_1, \ldots, B_n \vdash C}{\mathcal{X} : \Gamma, \Delta_1, \ldots, \Delta_n \vdash C}\ MC \qquad \frac{}{\mathcal{X} : A \vdash A}\ \text{Ax}$$

Figure 2 The multicut and axiom rules, where A denotes an atomic formula.

The structural rules include the usual complement: contraction, which builds in the treatment of multisets as sets, weakening, which allows us to add an extraneous assumption, an axiom rule which allows us to match an atomic conclusion with an assumption set comprising just that formula, and the cut rule that underlies the use of lemmas. We use a particular version of the cut rule called *multicut* that is better suited to our cut-elimination and consistency arguments. The axiom rule and the multicut rule are shown in Figure 2.

If an inference rule is a right introduction rule, all of its premises are called *major* premises. If it is a left introduction rule or a multicut, then only those premises with the same consequent as its conclusion are called major premises. All other premises are called *minor* premises.

2.3 Interpreting Atomic Formulas through Fixed-Point Definitions

The logic under consideration deviates from a conventional first-order logic in that it is parameterized by *fixed-point definitions* which provide the basis for analyzing atomic predicates. A fixed-point definition in this context is a set of clauses of the form $p\ \vec{t} \equiv_{\mathcal{X}} B$, where p is a predicate constant of type ω identified in the signature Σ, and where $(p\ \vec{t})$ and B are formulas lying over $\mathcal{X}$. Given such a clause $H \equiv_{\mathcal{X}} B$, we refer to H as its *head* and to B is its *body*. Following Tiu [18], we require every variable in $\mathcal{X}$ to appear in H, and H to belong to the higher-order pattern fragment of terms that is described, *e.g.*, in [15].

Let $\mathcal{D}$ be the fixed-point definition parameterizing the logic. Then $\mathcal{D}$ is intended to associate a meaning with each predicate symbol p in the following sense: an atomic formula $(p\ \vec{t})$ holds exactly when the formula matches the head of one of the clauses in $\mathcal{D}$ and the body of that clause holds. This interpretation is built into the logic by rules that allow an atomic conclusion of a sequent to be unfolded by one of the clauses whose head it instantiates and, conversely, an atomic assumption to be analyzed by all the possible clauses whose heads match it. Towards formalizing this understanding, letting $\mathcal{F}$ be a set of clauses of the kind we have described, we introduce the notation $\text{defn}_{\mathcal{F}}(H \equiv_{\mathcal{X}} B, A, \theta, B')$ to signify that there exists a clause $H \equiv_{\mathcal{X}} B \in \mathcal{F}$ and a substitution $\rho : \mathcal{Z} \to \mathcal{X}$ for some $\mathcal{Z}$, such that $H\rho = A\theta$ and $B' = B\rho$. Then the inference rules of the logic are augmented by the ones shown in Figure 3. Using the conventional terminology, these rules correspond, respectively to ones

for introducing atomic formulas on the left- and right-hand sides of sequents. Note that the set of premises in the $\Delta\mathcal{L}$ rule can be empty in particular instances, in which case the rule has the force of an axiom.

$$\frac{\{\text{range}(\theta) : \Gamma\theta, B' \vdash C\theta \mid \text{defn}_{\mathcal{D}}(H \equiv_{\mathcal{X}} B, p\ \vec{t}, \theta, B')\}}{\mathcal{Y} : \Gamma, p\ \vec{t} \vdash C}\ \Delta\mathcal{L}$$

$$\frac{\mathcal{Y} : \Gamma \vdash B'}{\mathcal{Y} : \Gamma \vdash p\ \vec{t}}\ \Delta\mathcal{R} \quad \text{defn}_{\mathcal{D}}(H \equiv_{\mathcal{X}} B, p\ \vec{t}, \epsilon_{\mathcal{Y}}, B')$$

Figure 3 Introduction rules for a fixed-point predicate p.

To illustrate fixed-point definitions, let us assume that the base types of the logic includes the type tm and that the signature Σ is the collection $\{z : tm, s : tm \to tm, even : tm \to \mathbf{o}\}$. Then the logic may be parameterized by a fixed-point definition comprising the clauses:

$$even\ z \equiv_{\emptyset} \top \qquad even\ (s\ (s\ X)) \equiv_X even\ X$$

Let us write $s^n\ z$ to denote the term $s\ (\ldots\ (s\ z))$ in which there are n occurrences of s; by an abuse of notation, $s^0\ z$ denotes z. Then this definition is intended to identify the collection $\{even\ (s^n\ z) \mid n \text{ is even}\}$ of atoms, i.e. this collection is its only fixed-point. This fact is manifest in the logic by the derivability of any sequent of the form $\cdot : \cdot \vdash even\ (s^n\ z)$ if n is even and, conversely, the derivability of any sequent of the form $\cdot : \cdot \vdash even\ (s^n\ z) \supset \bot$ if n is odd. Sequents of the former kind can be derived by a repeated use of the $\Delta\mathcal{R}$ rule and sequents of the latter kind can be derived by repeated uses of the $\Delta\mathcal{L}$ rule followed by a rule that introduces an implication on the right of the sequent.

Clauses in fixed-point definitions must be constrained to ensure that the logic is consistent. The constraint that we will use, which also accords well with the applications we envisage, is that of *ground stratification* described in [18]. Central to describing this constraint is a process for assigning an ordinal to each ground formula F that is called its *level* and is designated by $\text{lvl}(F)$. This process starts with an assignment for each ground atomic formula and extends it to arbitrary formulas based on the following rules:

$$\text{lvl}(\bot) := \text{lvl}(\top) := 0$$
$$\text{lvl}(A \wedge B) := \text{lvl}(A \vee B) := \max(\text{lvl}(A), \text{lvl}(B))$$
$$\text{lvl}(A \supset B) := \max(\text{lvl}(A) + 1, \text{lvl}(B))$$
$$\text{lvl}(\forall_\alpha x.C) := \text{lvl}(\exists_\alpha x.C) := \sup\{\text{lvl}(C[t/x]_\emptyset) \mid t \in \text{ground}(\alpha)\}$$

We then say that a fixed-point definition $\mathcal{D}$ is *ground stratified* if there exists a level assignment to ground atomic formulas such that for every clause $H \equiv_{\mathcal{X}} B \in \mathcal{D}$, and for every $\mathcal{X}$-grounding substitution ρ, it is the case that $\text{lvl}(H\rho) \geq \text{lvl}(B\rho)$. The requirement is then that the definition $\mathcal{D}$ be ground stratified.

A particular variant of ground stratification is that where the fixed-point definition is required to be stratified under a level assignment to atomic formulas that depends only on their predicate heads. It is this less permissive version of stratification that governs definitions in the logic $\mathcal{G}$ [7] that underlies the Abella proof assistant [2].

3 Inductive Definitions

The treatment of definitions in the logic so far is consistent with any fixed-point being associated with predicates in the sense sketched in Section 2.3. There can sometimes be a benefit to building into the logic an ability to narrow this choice down to the least fixed-point. We discuss this matter relative to a restricted version of ground stratified fixed-point definitions in the first subsection below. The remaining subsections discuss the possibility of extending such a treatment to arbitrary ground stratified fixed-point definitions. The section concludes with a complete description of the logic LD^{μ^+} whose consistency and utility will be considered in the next two sections.

3.1 Strictly Stratified Definitions and Least Fixed-Points

Fixed-point definitions can sometimes have multiple interpretations. To illustrate this possibility assume that the signature Σ identifies a propositional symbol p and then let $\mathcal{D}$ be a fixed-point definition comprising the sole clause $p \equiv_\emptyset p$. This definition is consistent with two different interpretations for p, one in which it does not hold and another in which it does; these correspond, respectively, to the least and greatest fixed-points associated with the definition. If we were to pick the least fixed-point, then we should be able to show that the sequent $\cdot : \cdot \vdash p \supset \bot$ holds. However, the logic does not build in that choice and this sequent is therefore not provable.

The logic can be extended to allow for a least fixed-point treatment of definitions. Formally, this is realized by parameterizing it by what we will call an *inductive* definition in addition to a fixed-point one. We will denote the inductive definition by $\mathcal{I}$. Predicate symbols in the signature must be distinguished as being of either the fixed-point or inductive variety and clauses for them may then appear only in $\mathcal{D}$ or $\mathcal{I}$. In what follows, we will denote a clause defining a fixed-point predicate by $H \stackrel{\Delta}{=}_{\mathcal{X}} B$ and a clause defining an inductive predicate by $H \stackrel{\mu}{=}_{\mathcal{X}} B$. We further require that clauses in $\mathcal{D}$ and $\mathcal{I}$ are ground stratified under a common level assignment to atomic formulas. At this juncture, we assume that the level assignment to an atomic formula with an inductive predicate as its head depends only on its head, a condition that we refer to by saying that $\mathcal{I}$ is *strictly stratified*; our objective in the remainder of this paper will be to remove this restriction. The concrete impact of the described condition can be understood as follows: in a clause of the form $p\ \vec{t} \stackrel{\mu}{=}_{\mathcal{X}} B$ that pertains to an inductively defined predicate symbol p, it must be the case that p does not appear in the antecedent of a top-level implication in the formula B.

The distinction in the treatment of inductive definitions within the logic arises from a special rule for introducing atomic formulas that are in the assumption sets of sequents and that have inductive predicates as their heads. To present this rule, we first recast clauses in $\mathcal{I}$ that are of the form $p\ \vec{t} \stackrel{\mu}{=}_{\mathcal{X}} B$ into the form $p\ \vec{t} \stackrel{\mu}{=}_{\mathcal{X}} B'\ p$, where $(B'\ p)$ λ-converts to B and B' is a term with no occurrences of p. Note that such a recasting is always possible: essentially, the "operator" B' can be obtained by abstracting over all occurrences of p in B. Given such a recasting, the relevant rule is the following:[2]

$$\frac{\{\mathcal{X} : B\ \mathrm{S} \vdash \mathrm{S}\ \vec{t} \mid p\ \vec{t} \stackrel{\mu}{=}_{\mathcal{X}} B\ p \in \mathcal{I}\} \qquad \mathcal{Y} : \Gamma, \mathrm{S}\ \vec{s} \vdash C}{\mathcal{Y} : \Gamma, p\ \vec{s} \vdash C}\ \mu\mathcal{L}_{ss}$$

[2] The subscript on $\mu\mathcal{L}_{ss}$ has the purpose of distinguishing it from the actual induction rule in LD^{μ^+} that we introduce in Section 3.3 and that applies to arbitrary ground stratified inductive definitions.

In an application of this rule, the term S, which is referred to as its *inductive invariant*, must be a ground term of the same type as p. The $\Delta\mathcal{R}$ rule applies unchanged to clauses in $\mathcal{I}$. The resulting logic has been shown to be consistent in [11]. We also note that the $\Delta\mathcal{L}$ rule is admissible in the presence of the $\mu\mathcal{L}_{ss}$ rule for strictly stratified inductive definitions, *i.e.*, it can be added to the calculus without changing what is derivable. This fact has been established in [17].

To see that inductive definitions add the ability to reason about least fixed-points, we may consider the example from the beginning of this section. Let $\mathcal{I}$ be the set $\{p \stackrel{\mu}{=}_\emptyset p\}$. The only clause in this set may be rewritten as $p \stackrel{\mu}{=}_\emptyset B\ p$ where B is $\lambda p.\, p$. Picking $\bot$ as the inductive invariant, we can prove with the $\mu\mathcal{L}_{ss}$ rule the sequent $\cdot : p \vdash \bot$, from which we easily obtain a proof of the sequent $\cdot : \cdot \vdash p \supset \bot$. As a more interesting example, consider the predicate $even$ defined by the clauses presented in Section 2.3. Treating the collection of those clauses as a fixed-point definition, we can construct a proof of the formula $even\ t \supset even\ (s\ t) \supset \bot$ for any ground term t. However, even though the fixed-point rules identify a unique interpretation for $even$, we cannot construct a proof of the formula $\forall x.\, even\ x \supset even\ (s\ x) \supset \bot$. This situation changes if we designate the definition to be an inductive one. The formula would hold if we can provide a derivation for the sequent $x : even\ x \vdash even\ (s\ x) \supset \bot$, which we can do using the $\mu\mathcal{L}_{ss}$ rule relative to $even$ with $\lambda x.\, even\ (s\ x) \supset \bot$ as the inductive invariant.

3.2 Unrestricted Ground Stratified Inductive Definitions

We now wish to extend our considerations to ground stratified inductive definitions without the strict stratification restriction. Towards this end, we first discuss how such definitions allow us to identify least fixed-points.

The usual way of constructing a least fixed-point for an operator Φ consists of collecting all possible pre-fixed-points, *i.e.*, sets X such that $\Phi(X) \subseteq X$, and then taking their intersection. If the operator is *monotone*, we can show that this intersection will be its least fixed-point. The principal hurdle we encounter with ground stratified definitions is that the operator we might naturally associate with them are not guaranteed to be monotone. To see this, consider the fixed-point definition comprising the single clause $odd\ (s\ X) \stackrel{\Delta}{=}_X odd\ X \supset \bot$, where odd is a predicate constant of type $tm \to \mathbf{o}$ and z and s are the constants already introduced in Section 2.3. Informally, we consider the set of ground atomic formulas, and identify the operator associated with a fixed-point definition $\mathcal{D}$ as $\Phi_\mathcal{D}(\mathcal{F}) := \{A \mid \mathrm{defn}_\mathcal{D}(H \stackrel{\Delta}{=}_\mathcal{X} B, A, \epsilon_\emptyset, B') \text{ and } \mathcal{F} \vDash B'\}$, where $\mathcal{F} \vDash B'$ signifies that B' follows from the ground atomic formulas in $\mathcal{F}$. Then, we may observe that when $\mathcal{D}$ consists of the clause defining odd, $\Phi_\mathcal{D}$ is not monotone because $\emptyset \subseteq \{odd\ z\}$ but $\Phi_\mathcal{D}(\emptyset) = \{odd\ (s^n\ z) \mid n \geq 1\}$ is not a subset of $\Phi_\mathcal{D}(\{odd\ z\}) = \{odd\ (s^n\ z) \mid n > 1\}$.

The solution we propose is to instead make sense of a ground stratified inductive definition via a *family* of inductive definitions defined by *recursion* on the level of its arguments. Informally, we interpret a ground stratified inductive predicate $p : \omega$ as a family of predicates $\{p_\alpha : \omega\}_\alpha$ where each p_α is a least fixed-point capturing the definition of p restricted to those terms $\vec{u}$ such that $\mathrm{lvl}((p\ \vec{u})\rho) \leq \alpha$ for any grounding substitution ρ. The negatively occurring instances of p in the body of the clauses defining p_α are replaced with some p_β for $\beta < \alpha$.[3] Thus, the way in which we index this family depends on the way in which we choose to stratify the definition. It can now be seen that the natural operator corresponding to the resulting definition will be monotone. We can then use this operator to identify a least-fixed point with the original definition.

[3] This idea does not quite work when α is a limit ordinal; in that case we would need to replace the negatively occurring instances of the predicate by a new predicate that "collects" the definitions of all the p_β for $\beta < \alpha$. We omit the details in this informal discussion.

We illustrate the above idea by making sense of the odd predicate in this way. We first ground stratify $odd\ X$ by a natural number denoting the number of occurrences of s in the ground argument X. Then, we define for each $n \geq 0$ an inductive predicate $odd_n : tm \to \mathbf{o}$ which we will see corresponds to the restriction of the predicate odd to terms of level at most n, as well as a predicate $lvl_n : tm \to \mathbf{o}$ internalizing those terms of level at most n:

$$lvl_0\ z \stackrel{\Delta}{=}_{\emptyset} \top \qquad odd_{n+1}\ X \stackrel{\mu}{=}_X odd_n\ X$$
$$lvl_{n+1}\ (s\ X) \stackrel{\Delta}{=}_X lvl_n\ X \qquad odd_{n+1}\ (s\ X) \stackrel{\mu}{=}_X lvl_{n+1}\ (s\ X) \wedge (odd_n\ X \supset \bot)$$

We now note that we may recast each odd_n as $odd_n\ (s\ X) \stackrel{\mu}{=}_X (B_n\ odd_n\ X)$, where each operator B_n is defined as follows

$$B_0 := \lambda S.\lambda x.\bot \quad \text{and} \quad B_{n+1} := \lambda S.\lambda x.(lvl_{n+1}\ (s\ x)) \wedge (odd_n\ x \supset \bot).$$

When viewed as transformers of sets of atoms, these operators are easily seen to be monotone.

Given such a family of operators, we can then identify a new predicate denoting the suprema, or infinitary disjunction of each of the operators, and this predicate identifies a least fixed-point. In our example, we define the predicate $odd_\omega : tm \to \mathbf{o}$, as well as a predicate $lvl_\omega : tm \to \mathbf{o}$ denoting that a term has a finite level, as the following infinite family of clauses

$$lvl_\omega\ X \stackrel{\Delta}{=}_X lvl_n\ X \qquad odd_\omega\ (s\ X) \stackrel{\mu}{=}_X B_n\ odd_\omega\ X$$

for each $n \geq 1$. We note that odd_ω cannot occur negatively in $(B_n\ odd_\omega\ x)$ for any n. From this it follows easily that these operators are, once again, monotone and we can therefore extract a least-fixed point, which we treat as the least-fixed point of the original definition. We observe in Lemma 11 of Appendix A that S is an inductive invariant for odd_ω if and only if S is an inductive invariant of odd_n for every n.

We could think of using the $\mu\mathcal{L}_{ss}$ rule in conjunction with the definition of odd_ω. The issue, however, is that odd_ω is an infinitary definition, making it impractical to work with. In the next section, we will describe an operator which allows us to approximate in a finitary manner what it means to be an inductive invariant for odd_ω in terms of odd, which has a finitary definition.

3.3 Inference Rules for Ground Stratified Inductive Definitions

To support arbitrary ground stratified inductive definitions, the logic needs to be equipped with inference rules that apply to them. As we have previously noted, the logic that results from extending the one described in Section 2 with strictly stratified inductive definitions complemented with the $\mu\mathcal{L}_{ss}$ and the $\Delta\mathcal{R}$ rule is consistent. However, using this set of rules with arbitrarily ground stratified inductive definitions renders the logic inconsistent. To see this, let us add to the example signature in Section 2.3 the predicate constants nat and odd of type $tm \to \mathbf{o}$ and let the inductive definition $\mathcal{I}$ comprise the clauses

$$nat\ z \stackrel{\mu}{=}_{\emptyset} \top \qquad even\ z \stackrel{\mu}{=}_{\emptyset} \top \qquad odd\ (s\ X) \stackrel{\mu}{=}_X nat\ X \wedge (odd\ X \supset \bot)$$
$$nat\ (s\ X) \stackrel{\mu}{=}_X nat\ X \qquad even\ (s\ (s\ X)) \stackrel{\mu}{=}_X even\ X$$

In this setting, we can construct derivations for the sequents $x : odd\ x \vdash even\ x \supset \bot$ and $x : odd\ x \vdash even\ x$ using the $\mu\mathcal{L}_{ss}$ rule with respect to odd; the argument in the first case will rely on the derivability of $\forall x.\, even\ (s\ x) \supset (even\ x \supset \bot)$ which follows easily from an observation made in Section 3.1; and the argument in the second case relies on the derivability of the sequent $x : nat\ x \vdash (even\ x \supset \bot) \supset even\ (s\ x)$, which we can construct by using the $\mu\mathcal{L}_{ss}$ rule with respect to the predicate nat. But then it follows that the sequent

$x : \mathit{odd}\ x \vdash \bot$ has a derivation. Instantiating x with z, we get a derivation of $\cdot : \cdot \vdash \mathit{odd}\ z \supset \bot$ and then, by the $\Delta\mathcal{R}$ rule, of $\cdot : \cdot \vdash \mathit{odd}\ (s\ z)$. By another instantiation, we see that there must also be a proof of $\cdot : \mathit{odd}\ (s\ z) \vdash \bot$. It then follows easily that there is a derivation of the sequent $\cdot : \cdot \vdash \bot$.

From a closer examination of the example above, we can see that the problem with using the $\mu\mathcal{L}_{ss}$ rule with ground stratified inductive definition is that the operators in the clauses defining a predicate could be abstracting over negative occurrences of the predicate as well, yielding an operator that is not monotone. To overcome this difficulty, we consider an alternative operator that "freezes" the definition in the negative context but abstracts it in the positive context. Specifically, for a fixed predicate constant $p : \omega$, we define by induction on the formula B the term $B^+_p : \omega \to \mathbf{o}$ as follows:

$$A^+_p := \lambda S. \begin{cases} S\ \vec{t} & \text{if } A = p\ \vec{t} \\ A & \text{if } A \text{ is } \top, \bot, \text{ or an atomic formula with a head different from } p \end{cases}$$

$$(B \wedge C)^+_p := \lambda S.(B^+_p\ S) \wedge (C^+_p\ S) \qquad (B \vee C)^+_p := \lambda S.(B^+_p\ S) \vee (C^+_p\ S)$$

$$(B \supset C)^+_p := \lambda S.B \supset (C^+_p\ S)$$

$$(\forall x.B)^+_p := \lambda S.\forall x.(B^+_p\ S) \qquad (\exists x.B)^+_p := \lambda S.\exists x.(B^+_p\ S)$$

We note that if B lies over $\mathcal{X}$, then so does B^+. We now recast the clauses in $\mathcal{I}$ that are of the form $p\ \vec{t} \stackrel{\mu}{=}_{\mathcal{X}} B$ into the form $p\ \vec{t} \stackrel{\mu}{=}_{\mathcal{X}} B^+_p\ p$. Then the rule for introducing atomic formulas with inductive predicate symbols as their heads is the following:

$$\frac{\{\mathcal{X} : B^+_p\ \mathrm{S} \vdash \mathrm{S}\ \vec{t} \mid p\ \vec{t} \stackrel{\mu}{=}_{\mathcal{X}} B \in \mathcal{I}\} \qquad \mathcal{Y} : \Gamma, \mathrm{S}\ \vec{s} \vdash C}{\mathcal{Y} : \Gamma, p\ \vec{s} \vdash C}\ \mu\mathcal{L}$$

As with the $\mu\mathcal{L}_{ss}$ rule, S must be a ground term of the same type as p and we refer to it as an inductive invariant for p.

We now observe that this new induction rule allows us to relate odd with odd_ω from Section 3.2. In particular, the relationship is that when restricted to arguments with a fixed level, the two definitions coincide. More precisely we show in Theorem 12 of Appendix A that for any x, $\mathit{odd}_\omega\ x$ if and only if $\mathit{odd}\ x \wedge \mathit{lvl}_\omega\ x$. In other words, the new induction rule over-approximates the least fixed-point odd_ω in the sense that odd may still hold on arguments which do not have a fixed level. We thus adopt the view that odd_ω is the predicate we are interested in defining, and that odd is merely an approximation of this predicate.

3.4 The Complete Logic

We can now identify LD^{μ^+} completely: it is the logic that extends the one described in Section 2 by parameterizing it additionally with an inductive definition $\mathcal{I}$ that is ground stratified by the same level assignment that $\mathcal{D}$ is and that is interpreted via the rule $\mu\mathcal{L}$ and a rule for introducing atomic formulas in the conclusion of a sequent that is identical to $\Delta\mathcal{R}$ except that it works over clauses in $\mathcal{I}$. For clarity, we shall refer to this rule as $\mu\mathcal{R}$. We observe that LD^{μ^+} is conservative with respect to the logic with ground stratified fixed-point definitions and strictly stratified inductive definitions interpreted by the $\mu\mathcal{L}_{ss}$ and $\mu\mathcal{R}$ rules: strict stratification trivially implies ground stratification and the $\mu\mathcal{L}_{ss}$ and $\mu\mathcal{L}$ rules are equivalent in its context.

4 Consistency of the Logic

Our objective in this section is to establish the consistency of LD^{μ^+}. A common approach to proving this property for a logic is to show that the cut rule is redundant to it, *i.e.*, to show that this rule can be *eliminated* from the calculus without impacting what is derivable;

an analysis of cut-free proofs then quickly shows that the sequent $\cdot : \cdot \vdash \bot$ is not derivable. However, a cut-elimination result has been difficult to prove for ground stratified versions of the logic of definitions. We circumvent this difficulty by using an indirect approach introduced by Tiu [18] to show the non-derivability of $\cdot : \cdot \vdash \bot$. We first describe a ground version of LD^{μ^+}, which we call $\mathrm{LD}^{\mu^+}_{\infty}$. We then show that cut elimination holds for $\mathrm{LD}^{\mu^+}_{\infty}$, which allows us to conclude that $\cdot : \cdot \vdash \bot$ is not derivable in $\mathrm{LD}^{\mu^+}_{\infty}$. Finally, we show that that if $\cdot : \cdot \vdash \bot$ is derivable in LD^{μ^+}, then it must also be derivable in $\mathrm{LD}^{\mu^+}_{\infty}$.

4.1 A Ground Version of the Logic

The formulas of $\mathrm{LD}^{\mu^+}_{\infty}$ are those of LD^{μ^+} which are well formed over the empty variable context. Sequents in $\mathrm{LD}^{\mu^+}_{\infty}$ are like sequents in LD^{μ^+} with the proviso that formulas in them are ground and their eigenvariable sets are empty. In light of the last fact, we omit mention of eigenvariables in sequents, writing them simply as $\Gamma \vdash C$. As in LD^{μ^+}, the predicate constants in the signature of $\mathrm{LD}^{\mu^+}_{\infty}$ are distinguished as being of the fixed-point or inductive variety and the logic is parameterized by a fixed-point definition $\mathcal{D}$ and an inductive definition $\mathcal{I}$ that have an identical shape to those in LD^{μ^+}.

$\mathrm{LD}^{\mu^+}_{\infty}$ has rules that exactly mirror the ones in LD^{μ^+}. All the rules in LD^{μ^+} except for $\forall\mathcal{R}$, $\exists\mathcal{L}$, and $\mu\mathcal{L}$, which we describe below, carry over unchanged to $\mathrm{LD}^{\mu^+}_{\infty}$. To distinguish them from those in LD^{μ^+}, we add ∞ as a subscript to their names. It is easily seen that all these rules other than perhaps the $\Delta\mathcal{L}$ rule translate into ones over $\mathrm{LD}^{\mu^+}_{\infty}$ sequents, *i.e.*, they are rules in which the sequents comprise only ground formulas. For the $\Delta\mathcal{L}$ rule that is reproduced below,

$$\frac{\{\Gamma, B' \vdash C \mid \mathrm{defn}_{\mathcal{D}}(H \stackrel{\Delta}{=}_{\mathcal{X}} B, A, \epsilon_{\emptyset}, B')\}}{\Gamma, A \vdash C}\ \Delta\mathcal{L}$$

we observe the following. If A is a ground formula, $H \stackrel{\Delta}{=}_{\mathcal{X}} B$ is a clause, and ρ is a substitution such that such $A = A\epsilon = H\rho$, then $B\rho$ must be a ground formula. This is a consequence of H being in the higher-order pattern fragment: all the substitutions ρ for the variables in H that unify it with a ground formula must be ground, and all the free variables in B are contained within this collection. It follows from this that all the premise sequents for this rule must be ground.

$$\frac{\{\Gamma \vdash C[t/x]_{\emptyset}\}_{t \in \mathrm{ground}(\alpha)}}{\Gamma \vdash \forall_{\alpha} x.C}\ \forall\mathcal{R}_{\infty} \qquad \frac{\{\Gamma, B[t/x]_{\emptyset} \vdash C\}_{t \in \mathrm{ground}(\alpha)}}{\Gamma, \exists_{\alpha} x.B \vdash C}\ \exists\mathcal{L}_{\infty}$$

$$\frac{\left\{B'^{+}_{p}\ \mathrm{S} \vdash \mathrm{S}\ \vec{u} \;\middle|\; \mathrm{defn}_{\mathcal{I}}(H \stackrel{\mu}{=}_{\mathcal{X}} B, p\ \vec{u}, \epsilon_{\emptyset}, B') \text{ and } \vec{u} \text{ is ground}\right\} \qquad \Gamma, \mathrm{S}\ \vec{t} \vdash C}{\Gamma, p\ \vec{t} \vdash C}\ \mu\mathcal{L}_{\infty}$$

Figure 4 The $\forall\mathcal{R}_{\infty}$, $\exists\mathcal{L}_{\infty}$, and $\mu\mathcal{L}_{\infty}$ rules in $\mathrm{LD}^{\mu^+}_{\infty}$.

The $\forall\mathcal{R}_{\infty}$, $\exists\mathcal{L}_{\infty}$, and $\mu\mathcal{L}_{\infty}$ rules are presented in Figure 4. The $\mu\mathcal{L}_{\infty}$ rule has the proviso that p must be an inductive predicate. As with the $\mu\mathcal{L}$ rule, S is expected to be a ground term of the same type as p, and we will refer to it as a *ground inductive invariant* for p. Given a ground inductive invariant S, with a family of derivations $\Pi_{\mathrm{S}}{}^{\vec{u},B'}$ for the sequent $B'^{+}_{p}\ \mathrm{S} \vdash \mathrm{S}\ \vec{u}$, we will denote the family of derivations $\{\Pi_{\mathrm{S}}^{\vec{u},B'}\}_{\vec{u},B'}$ by Π_{S} and call it a *witness set* for the invariant S.

We must verify that these new rules are well-defined over ground sequents. This is obviously true for the $\forall\mathcal{R}_\infty$ and $\exists\mathcal{L}_\infty$ rules. For the $\mu\mathcal{L}_\infty$ rule, we observe that if B is a ground formula then B_p^+ must be a ground term. That the rule is well-formed now follows from an argument similar to that used for $\Delta\mathcal{L}$ and the proviso that S must be a closed term. These rules are intended to be ground, infinitary versions of their counterparts in LD^{μ^+}. This relationship is evident for the $\forall\mathcal{R}_\infty$ and $\exists\mathcal{L}_\infty$ rules. For the $\mu\mathcal{L}_\infty$ rule, we observe that there is a sequence of ground terms $\vec{u}$ such that $\text{defn}_\mathcal{I}(p\ t \stackrel{\mu}{=}_\mathcal{X} B, p\ \vec{u}, \epsilon_\emptyset, B')$ exactly when there is a unique $\mathcal{X}$-grounding substitution $\eta : \emptyset \to \mathcal{X}$ such that $p\ \vec{u} = (p\ \vec{t})\eta$ and $B\eta = B'$.

4.2 Consistency of the Ground Logic

We now show that $\text{LD}_\infty^{\mu^+}$ is consistent by proving the cut-elimination theorem for it. The proof uses the reducibility technique, originally described by Tait and Martin-Löf (see [16, 13]) and adapted to the logic of definitions by McDowell and Miller [14]. The treatment of inductive definitions within this structure closely follows that in [17]. In substance, we first define a *reduction* relation, which specifies how the multicut rule may be propagated upwards in a derivation tree. Then, we introduce the notion of normalizable derivations, which allows us to show that the cut reduction relation is well-founded, and from which we can establish the cut-elimination theorem. To prove that every derivation is normalizable, we introduce the intermediate notion of a reducible derivation.

4.2.1 Cut Reductions

The reduction relation is a relation between derivations that is intended to *reduce* a derivation of the form

$$\dfrac{\begin{array}{cccc}\Pi_1 & & \Pi_n & \Pi \\ \Delta_1 \vdash B_1 & \cdots & \Delta_n \vdash B_n & B_1, \ldots, B_n, \Gamma \vdash C\end{array}}{\Delta_1, \ldots, \Delta_n, \Gamma \vdash C}\ MC$$

that ends with a multicut and that we denote by $mc(\Pi_1, \ldots, \Pi_n, \Pi)$ into another derivation. We refer to the formulas $B_1, \ldots, B_n$ in the multicut as *cut formulas*. If a left or right rule in the original derivation is not a contraction or weakening and it introduces a cut formula, we say that the rule is *principal*. If Π reduces to Π', we refer to Π as a *redex* and we call Π' a *reduct* of Π.

The reduction relation relates a derivation Ξ of the form shown to a new derivation Ξ' which may again end with a multicut but is always of smaller complexity. If $n = 0$, then Ξ reduces to Π. Otherwise, the reduction relation is specified by the case of the last rule of Π. If Π is principal, then we further group the reductions by cases on the derivation Π_i ending with the cut formula. Specifically, these are *essential cases* whenever Π_i also ends with a principal rule and Π does not end with $\mu\mathcal{L}_\infty$, an *inductive case* whenever Π ends with $\mu\mathcal{L}_\infty$, *left-commutative cases* whenever Π_i does not end with a principal rule, *structural cases* whenever Π ends with a weakening or contraction, *left axiom cases* whenever Π_i ends with an axiom, and *left multicut cases* whenever Π_i ends with a multicut. The remaining cases are when Π does not end with a principal rule. Here, we have the *right axiom case* when Π ends with an axiom, the *right multicut case* when it ends with a multicut, and the *right-commutative cases* for all other possibilities of the last rule in Π.

Except for the inductive case, the cut reduction relation that we use is identical to that described in [10].[4] Consequently, we discuss only the inductive case below, referring the reader to [10] for the other cases. In the inductive case, the reduction depends on a means of "unfolding" inductive definitions that is based on the following lemma; this lemma differs from the unfolding lemma in [11] in that p is not required to occur strictly positively in C.

▶ **Lemma 1** (Unfolding lemma). *Suppose* S *is a ground inductive invariant for an inductive predicate* p *with witness set* Π_{S}*, then for any derivation* Ξ *of a sequent* $\Delta \vdash C$*, there exists a derivation* $\mu(\Xi, \Pi_{\mathrm{S}})$ *of* $\Delta \vdash C^+_p\ \mathrm{S}$*.*

Proof. By induction on the height of the derivation Ξ. If Ξ ends with any rule (including a multicut) except $\supset\mathcal{R}$, Ax, or $\mu\mathcal{R}$, then $\mu(\Xi, \Pi_{\mathrm{S}})$ is obtained by unfolding each major premise derivation. We must treat specially the case in which Ξ ends with $\supset\mathcal{R}$, Ax, or $\mu\mathcal{R}$.

Case $\supset\mathcal{R}$**.** If Ξ ends with $\supset\mathcal{R}$, then the derivation $\mu(\Xi, \Pi_{\mathrm{S}})$ is obtained as follows

$$\dfrac{\begin{array}{c}\Xi' \\ \Delta, C_1 \vdash C_2\end{array}}{\Delta \vdash C_1 \supset C_2}\ \supset\mathcal{R} \qquad \rightsquigarrow \qquad \dfrac{\begin{array}{c}\mu(\Xi', \Pi_{\mathrm{S}}) \\ \Delta, C_1 \vdash C_{2_p}^+\ \mathrm{S}\end{array}}{\Delta \vdash C_1 \supset C_{2_p}^+\ \mathrm{S}}\ \supset\mathcal{R}$$

Case Ax. If Ξ ends with the axiom rule with atomic consequent distinct from p, then $\mu(\Xi, \Pi_{\mathrm{S}})$ is simply Ξ. Otherwise, if the axiom rule involves the atomic consequent $p\ \vec{t}$, then $\mu(\Xi, \Pi_{\mathrm{S}})$ is given by

$$\dfrac{\left\{ \begin{array}{c} \Pi_{\mathrm{S}}^{\vec{u}, B'} \\ B_p'^{+}\ \mathrm{S} \vdash \mathrm{S}\ \vec{u} \end{array} \;\middle|\; \mathrm{defn}_{\mathcal{I}}(H \stackrel{\mu}{=}_{\mathcal{X}} B, p\ \vec{u}, \epsilon_{\emptyset}, B') \text{ and } \vec{u} \text{ is ground} \right\} \qquad \dfrac{}{\mathrm{S}\ \vec{t} \vdash \mathrm{S}\ \vec{t}}\ \textsc{Init}}{p\ \vec{t} \vdash \mathrm{S}\ \vec{t}}\ \mu\mathcal{L}_\infty$$

Case $\mu\mathcal{R}$**.** If Ξ ends with $\mu\mathcal{R}$ for any predicate distinct from p, $\mu(\Xi, \Pi_{\mathrm{S}})$ is simply Ξ. If Ξ ends with $\mu\mathcal{R}$ for the inductive predicate p, then $\mu(\Xi, \Pi_{\mathrm{S}})$ is obtained as follows

$$\dfrac{\begin{array}{c}\Xi' \\ \Delta \vdash B'\end{array}}{\Delta \vdash p\ \vec{t}}\ \mu\mathcal{R} \qquad \rightsquigarrow \qquad \dfrac{\begin{array}{cc}\mu(\Xi', \Pi_{\mathrm{S}}) & \Pi_{\mathrm{S}}^{\vec{t}, B'} \\ \Delta \vdash B_p'^{+}\ \mathrm{S} & B_p'^{+}\ \mathrm{S} \vdash \mathrm{S}\ \vec{t}\end{array}}{\Delta \vdash \mathrm{S}\ \vec{t}}\ MC$$

◀

The reduction for the inductive case is then the following:

Essential inductive case: Let the redex be the derivation shown at the beginning of this subsection. An essential inductive case occurs when Π ends with a principal $\mu\mathcal{L}_\infty$ rule introducing the cut formula $p\ \vec{t}$ for an inductive predicate p. Without loss of generality, assume that the cut formula being introduced is $B_1 = p\ \vec{t}$. Furthermore, suppose Π has premise derivations Π_{S} and Π', where Π_{S} is a witness set for a ground inductive invariant S for p, and that Π' is a derivation of $\Gamma, \mathrm{S}\ \vec{t}, B_2, \ldots, B_n \vdash C$. Then the reduct is the following:

$$\dfrac{\begin{array}{cc}\mu(\Pi_1, \Pi_{\mathrm{S}}) & \left\{ \begin{array}{c} \Pi_j \\ \Delta_j \vdash B_j \end{array} \right\}_{j=2,\ldots,n} \qquad \Pi' \\ \Delta_1 \vdash \mathrm{S}\ \vec{t} & \qquad\qquad\qquad\qquad \Gamma, \mathrm{S}\ \vec{t}, B_2, \ldots, B_n \vdash C\end{array}}{\Delta_1, \ldots, \Delta_n, \Gamma \vdash C}\ MC$$

[4] There is, however, some eventually inconsequential differences to be mentioned that arise from the fact that [10] includes a treatment of generic quantification. Since such quantification is not included in LD^{μ^+} and therefore in $\mathrm{LD}^{\mu^+}_\infty$, we do not need to consider rules pertaining to it in the reductions and the sets of nominal constants associated with sequents are always empty.

4.2.2 Cut Elimination

Our aim is now to show how the cut reduction relation is well-founded, and can therefore be used to eliminate cuts from a derivation. We start with the following inductive definition, by which we mean the smallest set of derivations closed under the specified operation (see definition 1.1.1 in [1] for a precise definition).

▶ **Definition 2** (normalizability). *The set of normalizable derivations in $LD^{\mu^+}_{\infty}$ is inductively defined as follows:*

1. *if* Π *ends with a multicut, then* Π *is normalizable if every reduct* Π' *is normalizable*
2. *otherwise,* Π *is normalizable if each of its premise derivations is normalizable.*

By an induction on the definition of normalizability, it can be shown that a normalizable derivation must have a cut free proof. Thus, our goal becomes that of showing that every derivation in $\mathrm{LD}^{\mu^+}_{\infty}$ is normalizable. To do this we need the intermediate notion of γ-reducibility, where γ is an ordinal. We first identify the level $\mathrm{lvl}(\Pi)$ of a derivation Π of a (ground) sequent $\Gamma \vdash C$, with $\mathrm{lvl}(C)$. Then, γ-reducibility is inductively defined by transfinite recursion on the ordinal γ on derivations of level at most γ below.

▶ **Definition 3** (reducibility). *A derivation* Π *is* γ*-reducible if either it is* γ'*-reducible for* $\gamma' < \gamma$*, or* $\mathrm{lvl}(\Pi) = \gamma$ *and one of the following holds:*

1. *if* Π *ends with a multicut, then* Π *is* γ*-reducible if for every reduct* Π'*,* Π' *is* γ*-reducible.*
2. *if* Π *is the derivation*

$$\dfrac{\begin{array}{c}\Pi' \\ \Gamma, A \vdash B\end{array}}{\Gamma \vdash A \supset B}\,\supset\mathcal{R}\ ,$$

then Π *is* γ*-reducible if* Π' *is* γ*-reducible and for every* α*-reducible* $\begin{array}{c}\Upsilon \\ \Delta \vdash A\end{array}$ *where* $\alpha = \mathrm{lvl}(A)$*, the derivation* $mc(\Upsilon, \Pi')$ *is* γ*-reducible.*

3. *if* Π *ends with any other rule,* Π *is* γ*-reducible if each of its major premise derivations is* γ*-reducible, and each of its minor premise derivations is normalizable.*

To see that γ-reducibility is well defined, we note that if Π reduces to Π', then $\mathrm{lvl}(\Pi) = \mathrm{lvl}(\Pi')$, and furthermore, that levels are defined so that every rule has a non-increasing consequent in each of its major premises. In the case of the $\Delta\mathcal{R}$ and $\mu\mathcal{R}$ rules, this condition is guaranteed by the ground stratification condition. In all other cases, this condition is easily seen to be guaranteed by the definition of the level. Finally, we note that γ-reducibility of a derivation ending with the $\supset\mathcal{R}$ rule depends on α-reducibility to already be defined for any $\alpha < \gamma$.

We say a derivation Π is *reducible* if there exists a γ such that Π is γ-reducible. The lemma below asserts that reducibility implies normalizability. Since reducibility is a strengthening of normalizability, it is shown by straightforward induction on the γ-reducibility of a derivation.

▶ **Lemma 4** (Normalization Lemma). *If* Ξ *is* γ*-reducible then* Ξ *is normalizable.*

The key lemma needed to prove normalizability of derivations in $\mathrm{LD}^{\mu^+}_{\infty}$ is the following

▶ **Lemma 5** (Reducibility Lemma). *Given* n *derivations* $\Pi_1, \ldots, \Pi_n$ *such that* Π_i *is* γ_i*-reducible, and any derivation* Π*, the multicut*

$$\dfrac{\begin{array}{c}\Pi_1 \\ \Delta_1 \vdash B_1\end{array} \quad \cdots \quad \begin{array}{c}\Pi_n \\ \Delta_n \vdash B_n\end{array} \qquad \begin{array}{c}\Pi \\ \Gamma, B_1, \ldots, B_n \vdash C\end{array}}{\Delta_1, \ldots, \Delta_n, \Gamma \vdash C}\,MC$$

which we denote by Ξ*, is reducible.*

The proof follows the structure of that in [10]. The principal difference lies in the treatment of the essential induction case, which is dealt with by a straightforward adaptation of the reducibility of unfolded derivations lemma.

The cut-elimination theorem follows easily from Lemma 5 as we see below.

▶ **Theorem 6** (Cut admissibility for $\mathrm{LD}^{\mu^+}_{\infty}$). *There is a cut-free derivation in $LD^{\mu^+}_{\infty}$ for every ground sequent $\Gamma \vdash C$.*

Proof. Let Π be a derivation for $\Gamma \vdash C$. By Lemma 5, $mc(\Pi)$ is reducible. Since $mc(\Pi)$ reduces to Π, it follows that Π is reducible. Hence, by Lemma 4, Π must be normalizable. We now only need to note, by a straightforward induction on normalizability, that every normalizable derivation has a cut-free derivation. ◀

Since there can be no cut-free derivation for $\cdot \vdash \bot$ in $\mathrm{LD}^{\mu^+}_{\infty}$, we get the following corollary:

▶ **Corollary 7** (Consistency of $\mathrm{LD}^{\mu^+}_{\infty}$). *The sequent $\cdot \vdash \bot$ is not derivable in $LD^{\mu^+}_{\infty}$.*

4.3 Consistency of the Full Logic

The crux of our argument is to interpret a sequent of the form $\mathcal{Y} : \Gamma \vdash C$ in LD^{μ^+} by the family of ground sequents in $\mathrm{LD}^{\mu^+}_{\infty}$ that are obtained via the application of a $\mathcal{Y}$-grounding substitution. The following lemma relates an LD^{μ^+} sequent to its interpretation.

▶ **Lemma 8** (Grounding Lemma). *If $\mathcal{Y} : \Gamma \vdash C$ is derivable in LD^{μ^+}, then for any $\mathcal{Y}$-grounding substitution $\delta : \emptyset \to \mathcal{Y}$, the ground sequent $\Gamma\delta \vdash C\delta$ is derivable in $LD^{\mu^+}_{\infty}$.*

Proof. The proof is by induction on the height of the derivation of $\mathcal{Y} : \Gamma \vdash C$, and by case analysis on the last rule of the derivation. We only show the cases where the derivation ends with $\Delta\mathcal{L}$ or $\mu\mathcal{L}$ below.

Case $\Delta\mathcal{L}$. Let $\Gamma = \Gamma', A$ and suppose that the derivation ends with the following rule:

$$\frac{\{\mathrm{range}(\theta) : \Gamma'\theta, B' \vdash C\theta \mid \mathrm{defn}_{\mathcal{D}}(H \stackrel{\Delta}{=}_{\mathcal{X}} B, A, \theta, B')\}}{\mathcal{Y} : \Gamma', A \vdash C}\ \Delta\mathcal{L}$$

Observe first that $\mathrm{defn}_{\mathcal{D}}(H \stackrel{\Delta}{=}_{\mathcal{X}} B, A\delta, \epsilon_\emptyset, B')$ holds only if $\mathrm{defn}_{\mathcal{D}}(H \stackrel{\Delta}{=}_{\mathcal{X}} B, A, \delta, B')$ does. Thus, the premises of the indicated rule must include the sequent $\cdot : \Gamma'\delta, B' \vdash C\delta$ for each relevant B'. By the induction hypothesis, there must be an $\mathrm{LD}^{\mu^+}_{\infty}$ derivation for each such sequent. But then we can obtain an $\mathrm{LD}^{\mu^+}_{\infty}$ derivation for $\Gamma\delta \vdash C\delta$ by using the following rule:

$$\frac{\{\Gamma'\delta, B' \vdash C\delta \mid \mathrm{defn}_{\mathcal{D}}(H \stackrel{\Delta}{=}_{\mathcal{X}} B, A\delta, \epsilon_\emptyset, B')\}}{\Gamma'\delta, A\delta \vdash C\delta}\ \Delta\mathcal{L}$$

Case $\mu\mathcal{L}$. Let $\Gamma = \Gamma', p\ \vec{s}$ and suppose that the derivation ends with the following rule:

$$\frac{\{\mathcal{X} : B^+_p\ \mathrm{S} \vdash \mathrm{S}\ \vec{t} \mid p\ \vec{t} \stackrel{\mu}{=}_{\mathcal{X}} B \in \mathcal{I}\} \qquad \mathcal{Y} : \Gamma', \mathrm{S}\ \vec{s} \vdash C}{\mathcal{Y} : \Gamma', p\ \vec{s} \vdash C}\ \mu\mathcal{L}$$

As previously noted, for each sequence of ground terms $\vec{u}$ such that $\mathrm{defn}_{\mathcal{I}}(p\ \vec{t} \stackrel{\mu}{=}_{\mathcal{X}} B, p\ \vec{u}, \epsilon_\emptyset, B')$ holds, there is a unique $\mathcal{X}$-grounding substitution $\eta : \emptyset \to \mathcal{X}$ such that $p\ \vec{u} = (p\ \vec{t})\eta$ and $B\eta = B'$. But then, noting further that S is a ground term and using the induction hypothesis relative to the appropriate sequent in the collection of left premises for the rule shown, we

see that there must be an $\mathrm{LD}_\infty^{\mu^+}$ derivation for $B'^{+}_p\ \mathrm{S} \vdash \mathrm{S}\ \vec{u}$. Using the induction hypothesis again with the right premise of the rule, we see that the sequent $\Gamma'\delta, (\mathrm{S}\ \vec{s})\delta \vdash C\delta$ must have an $\mathrm{LD}_\infty^{\mu^+}$ derivation. Using these observations, we see that we may use an $\mu\mathcal{L}_\infty$ rule to get an $\mathrm{LD}_\infty^{\mu^+}$ derivation for the sequent $\Gamma\delta \vdash C\delta$. ◀

The following theorem is an easy consequence of Corollary 7 and Lemma 8.

▶ **Theorem 9** (Consistency of LD^{μ^+}). *LD^{μ^+} is consistent.*

5 An Example Application

We exhibit the use of ground stratified inductive definitions by considering aspects of an argument of strong normalizability for the simply typed lambda calculus. Such proofs typically identify a *reducibility predicate* that is shown to satisfy three *reducibility criteria* called CR1, CR2, and CR3. In the usual formulation of the reducibility predicate, these proofs proceed by induction on types [16, 8]. However, in our formulation, which is inspired by an idea from Martin-Löf's *computable deductions* [13], the proof of CR1 uses an induction on the reducibility predicate, which is a predicate whose definition is justified by ground stratification.

We use the types ty and tm and the following constants for representing types and terms:

$$\begin{array}{cccc} & \mathit{unit} : ty & \mathit{arr} : ty \to ty \to ty & \\ \mathit{star} : tm & \mathit{lam} : (tm \to tm) \to tm & \mathit{app} : tm \to tm \to tm & c : tm \end{array}$$

Here, star represents the unique term of type unit, and c is a unique constant, which may have any type, that functions as a stand-in for variables.[5] Note also that we are using a higher-order abstract syntax representation for lambda terms. We then need two predicates to encode steps in reductions and the notion of "neutrality" that recognizes variables and applications. For this we use the fixed-point predicate constants $\mathit{step} : tm \to tm \to \mathbf{o}$ and $\mathit{neutral} : tm \to \mathbf{o}$ that are defined by the following clauses:

$$\begin{array}{ll} \mathit{step}\ (\mathit{app}\ (\mathit{lam}\ S)\ U)\ (S\ U) \stackrel{\Delta}{=}_{S,U} \top & \mathit{neutral}\ c \stackrel{\Delta}{=}_{\emptyset} \top \\ \mathit{step}\ (\mathit{app}\ T\ U)\ (\mathit{app}\ T'\ U) \stackrel{\Delta}{=}_{T,T',U} \mathit{step}\ T\ T' & \mathit{neutral}\ (\mathit{app}\ T\ U) \stackrel{\Delta}{=}_{T,U} \top \\ \mathit{step}\ (\mathit{app}\ T\ U)\ (\mathit{app}\ T\ U') \stackrel{\Delta}{=}_{T,U,U'} \mathit{step}\ U\ U' & \\ \mathit{step}\ (\mathit{lam}\ S)\ (\mathit{lam}\ S') \stackrel{\Delta}{=}_{S,S'} \mathit{step}\ (S\ c)\ (S'\ c) & \end{array}$$

Finally, we encode the strong normalizability and reducibility predicates using the constants $\mathit{sn} : tm \to \mathbf{o}$ and $\mathit{red} : ty \to tm \to \mathbf{o}$ that are defined by the following inductive definition:

$$\begin{array}{l} \mathit{sn}\ T \stackrel{\mu}{=}_T \forall u.\mathit{step}\ T\ u \supset \mathit{sn}\ u \\ \\ \mathit{red}\ \mathit{unit}\ \mathit{star} \stackrel{\mu}{=}_{\emptyset} \top \\ \mathit{red}\ (\mathit{arr}\ A\ B)\ (\mathit{lam}\ S) \stackrel{\mu}{=}_{A,B,S} \forall u.(\mathit{red}\ A\ u) \supset (\mathit{red}\ B\ (S\ u)) \\ \mathit{red}\ B\ T \stackrel{\mu}{=}_{B,T} (\mathit{neutral}\ T) \wedge \forall t'.(\mathit{step}\ T\ t') \supset (\mathit{red}\ B\ t') \end{array}$$

These clauses are ground stratified by a measure that incorporates the size of the ground type A into the level of $\mathit{red}\ A\ T$. Similarly to the discussion in Section 3.2, we observe that we can define a family of predicates red_n defined on types of size n, from which we can then

[5] This treatment of variables suffices when proving CR1 but in order to prove CR2 we must use the usual apparatus of nominal constants to realize this functionality. We refer the reader to the Abella development that is provided in the supplementary material for details.

obtain a suprema red_ω. The proposition $red_\omega\ A\ T$ is what more closely corresponds to the mathematical statement that "T is reducible of type A", which analogously corresponds to the assertion that $typ\ A$ and $red\ A\ T$, where $typ\ A$ is a fixed-point asserting that A is a type.

Our interest in this example is in the proof of CR1, which is the property given by the formula $\forall t.\forall a.\ red\ a\ t \supset sn\ t$. We show this by proving the sequent $t, a : red\ a\ t \vdash sn\ t$, which we do by a use of the $\mu\mathcal{L}$ rule with respect to the predicate red. The inductive invariant that we use is $\lambda a.\,\lambda t.\ sn\ t$. The only premise for the rule with a non-trivial proof is that arising from the second clause where we will have to construct a proof for the sequent $a, s : \forall u. red\ a\ u \supset sn\ (s\ u) \vdash sn\ (lam\ s)$. Here we first observe that $red\ a\ c$ must be provable because c is neutral and does not step to any other term. The task then reduces to that of proving the sequent $s : sn\ (s\ c) \vdash sn\ (lam\ s)$. This can now easily be done with another application of the $\mu\mathcal{L}$ rule on the formula $sn\ (s\ c)$ using the inductive invariant $\lambda x.\exists y.(x = y\ c) \wedge sn\ (lam\ y)$.

Subsequent to showing that the criteria CR1, CR2, and CR3 are satisfied, the strong normalizability proof must show that every term satisfies the reducibility predicate. While this can be done by an induction on typing derivations in the present setting, this proof also requires an induction on the reducibility predicate in the case of Gödel's System T. Appendix B provides more details about this example.

6 Conclusion

In this paper, we have shown how to interpret ground stratified fixed-point definitions as inductive definitions. First, we have observed that we can make sense of such definitions as a family of inductive definitions identified by recursion on the level of the arguments of the predicate being defined. We have then introduced an induction rule via a new operator which approximates the infinitary interpretation of ground stratified inductive definitions as a family of least fixed-points and we have proved that the logic that includes this rule is consistent. The resulting logic subsumes the logic described in [11] that permitted only those inductive definitions that satisfied a stricter stratification condition. Finally, we have shown that the new rule allows us to encode a reducibility predicate that needs to be interpreted inductively, providing a new way of encoding strong normalizability proofs.

As we have noted already, the work in this paper as well as in [11] has built on that of Tiu in [18]. Another related effort is that of Baelde and Nadathur [3], which extended the deduction modulo framework of Dowek and Werner [5] with inductive and co-inductive definitions that are stratified by an ordering that is dependent only on the predicate name. A counterpart to this idea in the setting of type theory has also been developed by Jacob-Rao et al. [12]. In the logic called μNJ that is described in [3], rewrite rules can be used to realize fixed-point definitions with a flexibility that overlaps significantly with what is supported by ground stratification. Thus, μNJ provides capabilities that are akin to those supported by the logic described in [11] and this work extends them in similar ways. Another effort that is important to mention is that of Van den Eede et al. [19]. On the surface, the motivation and approach for this work is similar to ours: it too wants to ease the stratification restriction for inductive definitions, and it does this by introducing a proof rule like ours for introducing atomic formulas that appear as assumptions and that are defined inductively. However, the logics that are considered in the two cases are different. Van den Eede et al. work within a classical logic and they use the well-founded semantics from logic programming [20] in interpreting definitions. In contrast, we have worked within an intuitionistic logic, and our treatment of definitions and, consequently, our consistency arguments are entirely proof-

theoretic. Further, definitions are treated as formulas in [19] whereas in our setting they parameterize the logic. One consequence of this is that including a clause of the form $p \stackrel{\mu}{=}_{\emptyset} p \supset \bot$ in an inductive definition renders our logic inconsistent, whereas a similar clause can be included in the logic of Van den Eede et al. without having this effect. There are other differences too, such as in the way equality is understood in the two logics. Regardless of these differences, some of the new examples that can be treated by the two logics have a common flavor and it is of interest to understand the connections better.

We remark on three additional potential avenues for further research. First, while we believe that other features such as generic quantification, nominal abstraction and co-induction that are present in the logic $\mathcal{G}$ can co-exist naturally with ground stratified fixed-point and inductive definitions, we have not demonstrated this aspect in this work. We would like to do this in the near future so that the Abella proof system can be extended to incorporate our results with a theoretical guarantee of correctness. Second, the operator for our new induction rule allows us to replace only those occurrences of the predicate that are to the right of a top-level implication with an inductive invariant. We believe that it should be possible to consider a more inclusive operator that abstracts over all positive occurrences of the predicate.[6] We would like to verify the correctness of an induction rule that is based on such an operator and, additionally, we would like to understand the practical implications of permitting definitions that are stratified under the corresponding criterion. Finally, our interpretation of a ground stratified inductive definition as a family of inductive definitions defined by recursion on the level of its arguments is strikingly similar to inductive-recursive definitions in the type theory literature. In fact, it has been previously observed that the inductive method of reducibility in our example application corresponds to an inductive-recursive definition [6]. We would like to make this correspondence precise in future work.

References

1 Peter Aczel. An introduction to inductive definitions. In Jon Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, pages 739–782. Elsevier, 1977. doi:10.1016/S0049-237X(08)71120-0.

2 David Baelde, Kaustuv Chaudhuri, Andrew Gacek, Dale Miller, Gopalan Nadathur, Alwen Tiu, and Yuting Wang. Abella: A system for reasoning about relational specifications. *Journal of Formalized Reasoning*, 7(2), 2014. doi:10.6092/issn.1972-5787/4650.

3 David Baelde and Gopalan Nadathur. Combining deduction modulo and logics of fixed-point definitions. In *2012 27th Annual IEEE Symposium on Logic in Computer Science*, pages 105–114, 2012. doi:10.1109/LICS.2012.22.

4 Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5(2):56–68, 1940. doi:10.2307/2266170.

5 Gilles Dowek and Benjamin Werner. Proof normalization modulo. *Journal of Symbolic Logic*, 68(4):1289–1316, 2003. doi:10.2178/jsl/1067620188.

6 Peter Dybjer and Anton Setzer. Indexed induction–recursion. *The Journal of Logic and Algebraic Programming*, 66(1):1–49, 2006. doi:10.1016/j.jlap.2005.07.001.

7 Andrew Gacek, Dale Miller, and Gopalan Nadathur. Nominal abstraction. *Information and Computation*, 209(1):48–73, January 2011. doi:10.1016/j.ic.2010.09.004.

8 Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and Types*. Cambridge University Press, 1989.

9 Kurt Gödel. On a hitherto unexploited extension of the finitary standpoint. *Journal of Philosophical Logic*, 9:133–142, 1980. doi:10.1007/BF00247744.

[6] Such an operator is, in fact, used in the proof rule presented in [19].

10 Nathan Guermond. Towards weak stratification for logics of definitions, 2026. doi:10.48550/arXiv.2602.03072.

11 Nathan Guermond and Gopalan Nadathur. Ground stratification for a logic of definitions with induction. *Electronic Proceedings in Theoretical Computer Science*, 431:1–16, October 2025. doi:10.4204/eptcs.431.1.

12 Rohan Jacob-Rao, Brigitte Pientka, and David Thibodeau. Index-Stratified Types. In Hélène Kirchner, editor, *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018)*, volume 108 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSCD.2018.19.

13 Per Martin-Löf. Hauptsatz for the intuitionistic theory of iterated inductive definitions. In *Proc. of the Second Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and the Foundations of Mathematics*, pages 179–216. North-Holland, 1971. doi:10.1016/s0049-237x(08)70847-4.

14 Raymond McDowell and Dale Miller. Cut-elimination for a logic with definitions and induction. *Theoretical Computer Science*, 232:91–119, 2000. doi:10.1016/S0304-3975(99)00171-1.

15 Dale Miller and Gopalan Nadathur. *Programming with Higher-Order Logic*. Cambridge University Press, June 2012. doi:10.1017/CBO9781139021326.

16 W. W. Tait. Intensional interpretations of functionals of finite type I. *Journal of Symbolic Logic*, 32(2):198–212, 1967. doi:10.2307/2271658.

17 Alwen Tiu. *A Logical Framework for Reasoning about Logical Specifications*. PhD thesis, Pennsylvania State University, May 2004. URL: http://etda.libraries.psu.edu/theses/approved/WorldWideIndex/ETD-479/.

18 Alwen Tiu. Stratification in logics of definitions. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Automated Reasoning*, pages 544–558, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. doi:10.1007/978-3-642-31365-3_43.

19 Robbe Van den Eede, Robbe Van Biervliet, and Marc Denecker. A sequent calculus for generalized inductive definitions. In Carmine Dodaro, Gopal Gupta, and Maria Vanina Martinez, editors, *Logic Programming and Nonmonotonic Reasoning*, pages 30–42. Springer Nature Switzerland, 2025. doi:10.1007/978-3-031-74209-5_3.

20 Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3):619–649, July 1991. doi:10.1145/116825.116838.

A Least Fixed-Points for Ground Stratified Inductive Definitions

In Section 3.2, we defined a family of least fixed-points odd_n, as well as the suprema of this family odd_ω. We remarked that a predicate S is an inductive invariant of odd_ω if and only if it is an inductive invariant of each odd_n. We substantiate this claim in Lemma 11. Then, we claimed in Section 3.3 that the ground stratified inductive definition for odd could be related to odd_ω by Theorem 12, which holds in LD^{μ^+}. In this appendix, we prove this theorem.

For convenience, we recall the definitions of lvl_n and odd_n:

$$lvl_0\ z \stackrel{\Delta}{=}_{\emptyset} \top \qquad odd_{n+1}\ X \stackrel{\mu}{=}_X odd_n\ X$$
$$lvl_{n+1}\ (s\ X) \stackrel{\Delta}{=}_X lvl_n\ X \qquad odd_{n+1}\ (s\ X) \stackrel{\mu}{=}_X lvl_{n+1}\ (s\ X) \wedge (odd_n\ X \supset \bot)$$

We begin with the following lemma relating odd with odd_n, which is proved by a meta-level induction on the natural number n.

▶ **Lemma 10.** *The ground inductive definition odd is characterized in terms of the family of least fixed-point definitions odd_n as follows*

$$x : odd_n\ x \vdash odd\ x \quad \textit{and} \quad x : lvl_n\ x, odd\ x \vdash odd_n\ x.$$

Proof. If $n = 0$, we note that there are no clauses associated to odd_0, thus it trivially follows that odd is an inductive invariant for odd_0. Conversely, by case analysis on $lvl_0\ x$, x must be z, and it is easily shown that $\cdot : odd\ z \vdash odd_0\ z$ since $odd\ z$ may be proven false.

For the inductive case, we suppose the theorem holds for some n and we show that

$$x : odd_{n+1}\ x \vdash odd\ x \quad \text{and} \quad x : lvl_{n+1}\ x, odd\ x \vdash odd_{n+1}\ x.$$

Firstly, we show that odd is an inductive invariant of odd_{n+1}. This require showing firstly that $x : odd_n\ x \vdash odd\ x$, which is precisely the first component of the induction hypothesis, and secondly that $x : lvl_{n+1}\ (s\ x), odd_n\ x \supset \bot \vdash odd\ (s\ x)$. This is shown first by case analysis on $lvl_{n+1}\ (s\ x)$, and then then by unfolding $odd\ (s\ x)$ to obtain $x : lvl_n\ x, odd_n\ x \supset \bot \vdash odd\ x \supset \bot$. But this follows from the derivability of $x : lvl_n\ x, odd\ x \vdash odd_n\ x$, which is precisely the second component of the induction hypothesis.

Conversely, it suffices to show that $\lambda x.lvl_{n+1}\ x \supset odd_{n+1}\ x$ is an inductive invariant for $odd\ x$. We must show that $x : odd\ x \supset \bot \vdash lvl_{n+1}\ (s\ x) \supset odd_{n+1}\ (s\ x)$. To do this, we introduce the hypothesis, and unfold $odd_{n+1}\ (s\ x)$ to obtain $x : odd\ x \supset \bot, lvl_{n+1}\ (s\ x) \vdash odd_n\ x \supset \bot$. To see that this holds it suffices to observe that $x : odd_n\ x \vdash odd\ x$ from the first component of the induction hypothesis. ◀

We then prove the following lemma relating odd_ω with odd_n.

▶ **Lemma 11.** *A ground predicate* S *is an inductive invariant for* odd_ω *if and only if it is an inductive invariant for every* odd_n*.*

Proof. Suppose S is an inductive invariant for odd_ω, then S satisfies $x : B_n\ \mathrm{S}\ x \vdash \mathrm{S}\ (s\ x)$ for every n. We proceed by induction on n. Since odd_0 has no defining clauses, S is trivially an inductive invariant for it. To show that S is an inductive invariant for odd_{n+1}, we must show that $x : odd_n\ x \vdash \mathrm{S}\ x$, and that $x : B_{n+1}\ \mathrm{S}\ x \vdash \mathrm{S}\ (s\ x)$. The first condition follows from the induction hypothesis. The second condition is easily shown from our assumption about S.

The converse of the lemma is immediate. ◀

This allows us to conclude with the following theorem relating odd with odd_ω.

▶ **Theorem 12.** *On arguments with a fixed level,* odd *coincides with the least fixed-point* odd_ω*, that is,*

$$x : odd_\omega\ x \vdash odd\ x \quad \textit{and} \quad x : lvl_\omega\ x, odd\ x \vdash odd_\omega\ x.$$

Proof. To see that $x : odd_\omega\ x \vdash odd\ x$, it suffices to show that $odd\ x$ is an inductive invariant for odd_ω, but by Lemma 10, odd is an inductive invariant for each odd_n, and by Lemma 11, it follows that odd is an inductive invariant for odd_ω.

To see that $x : lvl_\omega\ x, odd\ x \vdash odd_\omega\ x$, it suffices to do case analysis on $lvl_\omega\ x$ and to show that for any n, $x : lvl_n\ x, odd\ x \vdash odd_\omega\ x$. To show this it is sufficient by Lemma 10 to show that $x : odd_n\ x \vdash odd_\omega\ x$, but this follows from the observation that odd_ω is an inductive invariant of itself, and is thus an inductive invariant of odd_n by Lemma 11. ◀

B Strong Normalizability for System T

As a more involved application of ground stratified inductive definitions, we consider a generalization of the argument of strong normalizability for the simply typed lambda calculus to the well-typed terms in System T [9]. Specifically, we describe the crux of a proof that uses an induction on the reducibility of a term to establish that the recursor is reducible if

all of its arguments are. In this formalization, we make use of generic quantification, nominal constants, and nominal abstraction to deal with free variables and variable bindings in the object language. The extension of the logic that we have presented in this paper with these additional features is the subject of a forthcoming paper. The formalization that we sketch has been mechanized in the Abella system, and is available as supplementary material.

We begin by introducing the constant symbols $nat : ty$ and $arr : ty \to ty \to ty$ used to construct terms of type ty encoding types. Then, we introduce a type of terms tm used to encode terms in System T constructed from the following constant symbols

$$z : tm \qquad s : tm \to tm \qquad lam : (tm \to tm) \to tm$$
$$rec : tm \to tm \to tm \to tm \qquad app : tm \to tm \to tm$$

We then define a relation $step : tm \to tm \to \mathbf{o}$ specifying a reduction relation on terms by first specifying the redexes

$$step\ (rec\ C\ F\ z)\ C \stackrel{\Delta}{=}_{C,F} \top \qquad step\ (rec\ C\ F\ (s\ U))\ (app\ (app\ F\ U)\ (rec\ C\ F\ U)) \stackrel{\Delta}{=}_{C,F,U} \top$$
$$step\ (app\ (lam\ S)\ U)\ (S\ U) \stackrel{\Delta}{=}_{S,U} \top \qquad step\ (lam\ S)\ (lam\ S') \stackrel{\Delta}{=}_{S,S'} \nabla x.step\ (S\ x)\ (S'\ x)$$

and then extending the $step$ relation by the clause $step\ (f\ \vec{u})\ (f\ \vec{u}') \stackrel{\Delta}{=}_{\vec{x}} step\ u_i\ u_i'$ for each constant symbol f (except for lam) of arity n, and for each $1 \leq i \leq n$. If $step\ U\ U'$, we say that U reduces to U', and sometimes write $U \to U'$. As before, we use the reduction relation to inductively define the strong normalizability predicate $sn : tm \to \mathbf{o}$, where we say that a term U is normalizable if it satisfies $sn\ U$.

Now, we may specify the reducibility predicate $red : ty \to tm \to \mathbf{o}$ by a ground stratified inductive definition

$$red\ nat\ z \stackrel{\mu}{=}_{\emptyset} \top \qquad red\ (arr\ A\ B)\ (lam\ S) \stackrel{\mu}{=}_{A,B,S} \forall u.(red\ A\ u) \supset (red\ B\ (S\ u))$$
$$red\ nat\ (s\ U) \stackrel{\mu}{=}_{U} red\ nat\ U \qquad red\ B\ T \stackrel{\mu}{=}_{B,T} (neutral\ T) \wedge \forall t'.(step\ T\ t') \supset (red\ B\ t')$$

where neutral terms are defined by the predicate $neutral : tm \to \mathbf{o}$ as follows

$$\nabla x.neutral\ x \stackrel{\Delta}{=}_{\emptyset} \top \qquad neutral\ (app\ T\ U) \stackrel{\Delta}{=}_{T,U} \top \qquad neutral\ (rec\ C\ F\ U) \stackrel{\Delta}{=}_{C,F,U} \top$$

We will say that a term U is reducible of type A if it satisfies $red\ A\ U$.

The proof is as follows. First, we prove the normalizability lemma (CR1), which asserts that reducible terms are normalizable. This is done analogously to CR1 in Section 5. Then, similarly to the proof in [8], (CR2) is proved by induction on the type A. We then introduce two lemmas, which assert that each of the neutral terms app and rec are reducible if each of their arguments are reducible. Similarly to the proof in [8], the reducibility of $(app\ T\ U)$ is shown by induction on the normalizability of T and U. The reducibility of the recursor is the key lemma which requires induction on the reducibility of a term. The proof in [8] makes use of induction on *the number of successors in the normal form of* U, which here is captured by our inductive definition of reducibility.

▶ **Lemma 13** (Reducibility of the recursor). *If C is reducible of type A, F is reducible of type $(arr\ nat\ (arr\ A\ A))$, and U is reducible of type nat, then $(rec\ C\ F\ U)$ is reducible of type A.*

Proof. The proof is by induction on the reducibility of U. By (CR1), we know that both C and F are normalizable. Thus we proceed by subordinate induction on both the normalizability of C and on the normalizability of F.

Since $rec\ C\ F\ U$ is neutral, we must show that each of its reducts is reducible of type A.

Case 1. Suppose U is z and that $rec\ C\ F\ z \to C$, then C is reducible of type A by our assumption.

Case 2a. Suppose U is $s\ U'$ and that $rec\ C\ F\ (s\ U') \to app\ (app\ F\ U')\ (rec\ C\ F\ U')$. Since U' is a predecessor of U, it follows from the induction hypothesis on the reducibility of U that $rec\ C\ F\ U'$ is reducible of type A. Then, since U' is reducible of type A by the definition of the reducibility of U, it follows by the reducibility of applications lemma and the reducibility of F that $app\ (app\ F\ U')\ (rec\ C\ F\ U')$ is reducible of type A.

Case 2b. Suppose that U is $s\ U'$ and that $rec\ C\ F\ (s\ U') \to rec\ C\ F\ (s\ U'')$ for some U'' such that $U' \to U''$. By (CR2), it suffices to show that $rec\ C\ F\ (s\ U')$ is reducible. This follows from a sublemma asserting that the reducibility of $rec\ C\ F\ U'$ implies the reducibility of $rec\ C\ F\ (s\ U')$, which is proved by induction on the normalizability of U', C, and F.

Case 3. Suppose U is neutral, and that $rec\ C\ F\ U \to rec\ C\ F\ U'$ for some U' such that $U \to U'$. Since U' is a predecessor of U, it follows from the induction hypothesis on the reducibility of U that $rec\ C\ F\ U'$ is reducible of type A.

Case 4. Suppose that $rec\ C\ F\ U \to rec\ C'\ F\ U$ for some C' such that $C \to C'$. Then it follows by the induction hypothesis on the normalizability of C, together with (CR2), that $rec\ C'\ F\ U$ is reducible of type A.

Case 5. The case in which $rec\ C\ F\ U \to rec\ C\ F'\ U$ for some F' such that $F \to F'$ follows an identical argument to that in Case 4. ◀

In the following lemma, we omit the definition of what it means to be a well typed term, as this is specified in the Abella development. The lemma is proved by a straightforward induction on the typing derivation of the term T. The introduction rules for the constants z, s and lam follow from the definition of reducibility, whereas the introduction rules for the neutral constants app and rec follow from the reducibility lemma for applications and Lemma 13 above.

▶ **Lemma 14** (Reducibility Lemma). *For any term $(T\ x_1\ \ldots\ x_n)$ of type B in context $x_1 : A_1, \ldots, x_n : A_n$, then for any sequence of terms $U_1, \ldots, U_n$ such that each U_i is reducible of type A_i, $(T\ U_1\ \ldots\ U_n)$ is reducible of type B.*

Since every well-typed term is reducible by the reducibility lemma, and every reducible term is normalizable, we obtain as a corollary

▶ **Corollary 15** (Strong normalizability). *Every well typed term is normalizable.*