Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Zyklische Superposition und Induktion

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Logic and Computation

eingereicht von

Jannik Vierling
Matrikelnummer 1226434

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dr.techn. Stefan Hetzl

Wien, 1. Februar 2018

Jannik Vierling	Stefan Hetzl

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Cyclic Superposition and Induction

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Logic and Computation

by

Jannik Vierling
Registration Number 1226434

to the Faculty of Informatics
at the TU Wien

Advisor: Associate Prof. Dipl.-Ing. Dr.techn. Stefan Hetzl

Vienna, 1st February, 2018

Jannik Vierling

Stefan Hetzl

Erklärung zur Verfassung der Arbeit

Jannik Vierling
Neulerchenfelder Straße, 17/2/44, 1660 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Februar 2018

Jannik Vierling

Acknowledgements

I would like to thank my advisor Stefan Hetzl for providing me with a very interesting topic, and for giving me the possibility to work in the area of proof theory and inductive theorem proving. I am also thankful to Stefan for his patience, his invaluable advice, and for the many inspiring discussions.

I would also like to thank my parents for making my studies possible.

I dedicate this work to Petra for her endless support.

Kurzfassung

Induktives Beweisen ist der Teilgebiet der automatischen Deduktion, das sich mit der Automatisierung des Beweisens mathematischer Aussagen mittels mathematischer Induktion befasst. Da Induktionsinvarianten im Allgemeinen nicht-analytisch sind, ist es schwierig effiziente Methoden zu entwickeln, die in der Lage sind Aussagen über einer gegebenen Theorie zu beweisen. Das Hauptproblem besteht darin Induktionsinvarianten zu berechnen, die ausreichen um eine gegeben Aussage zu beweisen.

Es wurden einige Ansätze für die Lösung dieses Problems vorgeschlagen. Unter anderem gibt es Ansätze die auf zyklischen Beweisen [KP13, BGP12], Baumgrammatiken [EH15], und Rippling basieren [BSVH$^+$93]. Der von Kersani und Peltier in [KP13] vorgeschlagene Ansatz besteht darin einen Superpositionskalkül um eine Zyklenerkennung zu erweitern. Die Zyklen in den Deduktionen des Superpositionskalküls stellen Argumentationen durch unendlichen Abstieg dar. Dieser Kalkül wird n-Klausel Kalkül genannt. Der n-Klausel Kalkül wurde nicht dafür entworfen um eine a priori bestimmte Art von Induktion zu erfassen. Daher ist bis heute nicht geklärt, welche Sätze durch diesen Kalkül bewiesen werden können. Allerdings wird vermutet, dass der n-Klausel Kalkül nur eine "schwache" Form der Induktion erfasst.

Das Ziel dieser Diplomarbeit ist es die Menge der im n-Klausel Kalkül beweisbaren Sätze zu untersuchen. Insbesondere sollen die im n-Klausel Kalkül beweisbaren Sätze durch die Quantorenkomplexität der in den induktiven Zyklen enthaltenen Induktionsinvarianten beschrieben werden. Mögliche Induktionsinvarianten werden anhand einer Zwei-Schritt Übersetzung ausfindig gemacht. Im ersten Schritt werden induktive Zyklen des n-Klausel Kalküls in den von Brotherston und Simpson in [BS10] vorgeschlagenen zyklischen Sequentialkalkül CLKID^ω übersetzt. Im zweiten Schritt werden die vorher gewonnenen Beweise des zyklischen Sequentialkalküls in Beweise des Sequentialkalküls mit Induktionsregel umgewandelt, wodurch die Induktionsinvarianten offengelegt werden. Die Induktionsinvarianten, die durch dieses Verfahren erhalten werden, sind immer Σ_1-Formeln. Daher stellt diese Diplomarbeit die obere Schranke der Σ_1-Induktion für die durch den n-Klausel Kalkül erfassten induktiven Argumente her.

Diese Diplomarbeit schließt mit einigen Intuitionen bezüglich der Optimalität dieser oberen Schranke, und der vollständigkeit des n-Klausel Kalküls hinsichtlich der Σ_1-Induktion ab.

Abstract

Inductive theorem proving is the branch of automated deduction that studies the automation of proving mathematical statements by means of mathematical induction. Because induction invariants are in general non-analytic it is difficult to devise efficient procedures which are able to prove statements over a given theory. The main issue is the computation of induction invariants which are sufficient to prove a statement at hand.

Several approaches have been proposed to deal with this problem. Among others there are approaches based on cyclic proofs [KP13, BGP12], tree-grammars [EH15], and rippling [BSVH$^+$93]. The approach proposed by Kersani and Peltier in [KP13] consists in enhancing a superposition calculus by a cycle detection rule. These cycles in superposition deductions represent arguments by infinite descent. This cyclic superposition calculus is called the n-clause calculus. The n-clause calculus was not designed to capture some a priori known type of induction. Therefore, it is until now unknown which sentences are provable in this calculus. Nevertheless it is suspected that the n-clause calculus captures only a "weak" notion of induction.

This thesis aims at investigating the set of sentences that are provable by the n-clause calculus. In particular, the sentences provable in n-clause calculus will described with with respect to the quantifier complexity of the induction invariants contained in the inductive cycles. Possible induction invariants are discovered by a two step translation. In a first step, inductive cycles of the n-clause calculus are translated to the cyclic sequent calculus CLKID^ω introduced by Brotherston and Simpson in [BS10]. In a second step, the proofs of the cyclic sequent calculus obtained in the first step, are translated to proofs of the sequent calculus with induction rules, thereby, revealing the induction invariants. The induction invariants discovered in this way are always Σ_1-formulas. Hence, this thesis establishes an upper bound of Σ_1-induction for the inductive arguments captured by the n-clause calculus.

The thesis concludes by giving some intuitions as to whether the obtained induction invariants are optimal with respect to their quantifier complexity, and whether the n-clause calculus is complete with respect to Σ_1-induction.

Contents

Kurzfassung **ix**

Abstract **xi**

Contents **xiii**

1 Introduction **1**
1.1 Induction and Infinite Descent 1
1.2 Inductive Theorem Proving 2
1.3 Problem Statement 3
1.4 Structure of the Thesis 4

2 Preliminary Definitions **7**
2.1 Term Languages 7

3 Kersani and Peltier's calculus **11**
3.1 The n-Clause Logic 11
3.2 The n-Clause Calculus 15

4 First-Order Logic with Inductive Definitions **23**
4.1 Syntax 23
4.2 Semantics 25
4.3 Sequent Calculi for $\mathrm{FOL}_{\mathrm{ID}}$ 29
4.4 The Brotherston-Simpson Conjecture 38

5 Translation of n-Clause Logic to $\mathrm{FOL}_{\mathrm{ID}}$ **41**
5.1 The Language 41
5.2 Constraint Clauses 42
5.3 Semantic Analysis 43

6 Translation of the n-Clause Calculus to LKID **51**
6.1 Clause Normalization 52
6.2 Superposition Deductions 54
6.3 Inductive Cycles 59

6.4 Simple Cyclic Proofs 63
6.5 Refutations of the n-Clause Calculus 67

7 Open Questions **73**
7.1 Optimality of the Induction Invariants 73
7.2 Incompleteness with respect to Σ_1-induction 75

8 Conclusion **81**

Bibliography **83**

CHAPTER 1

Introduction

In Section 1.1 we shall shortly introduce the principle of induction on the metalevel and the related principle of infinite descent. In Section 1.2 we will have a very brief introduction to inductive theorem proving. We will discuss the motivation for inductive theorem proving and why it is so difficult, moreover we shall discuss some of the most relevant applications of inductive theorem proving. Finally in Section 1.3 we will present the background and a more detailed outline of the problem addressed in this thesis.

1.1 Induction and Infinite Descent

In a set-theoretic context the principle of induction uses a well-ordering of a set to infer that a certain property that inherits from the smaller elements to larger elements eventually holds for all elements of that set. The principle of induction is based on the following observation.

Theorem 1.1. *Let A be a set, $<$ a well-ordering over A, and P a proposition about elements of A. If for all $x \in A$, we have $P(y)$ for all $y \in A$ with $y < x$, then we have $P(a)$ for all $a \in A$.*

Proof. We proceed by contradiction and assume that there exists an element $a \in A$ such that $P(a)$ does not hold. Let $A' = \{a \in A \mid P(a) \text{ does not hold}\}$ and let c be the minimal element of A' with respect to $<$. Since c is minimal in A', every element b in A with $b < a$ must satisfy $P(b)$. But this contradicts the assumptions. □

The induction principle that explicitly uses a well-order $<$ is called the principle of *order induction*. We can state this principle as follows:

$$(\forall x \in A : (\forall y \in A : y < x \to P(y)) \to P(x)) \to (\forall x \in A : P(x)).$$

Depending on how the set is constructed we may define variants the principle of induction which implicitly use the well-ordering induced by the construction of the set under

consideration. We then speak of *structural induction*. For example the natural numbers can be defined as the least set containing the number 0 and being closed under the operation $x \mapsto x + 1$. Hence, we can also carry out induction as follows. Let P be a property over the natural numbers. If P(0) holds and, for every number $x \in \mathbb{N}$, $P(x)$ implies $P(x + 1)$, then we conclude that $P(x)$ holds for every number $x \in \mathbb{N}$. We can state this principle more concisely as follows:

$$P(0) \wedge (\forall x \in \mathbb{N} : P(x) \rightarrow P(x + 1)) \rightarrow (\forall x \in \mathbb{N} : P(x)).$$

This principle is also sometimes referred to as the principle of weak induction. Note that this principle can be defined without having "access" to the natural well-ordering $<$ of the natural numbers.

Besides strong induction and weak induction other induction schemes such as j-step induction, j-induction, (i, j)-induction, and polynomial induction can be formulated for the natural numbers [WH17]. We will consider the principle of polynomial induction later in Chapter 7 when discussing some completeness properties of the n-clause calculus. Depending on the formal system, these principles may have different strength [BT17a, WH17].

A related notion is that of *infinite descent*. Indeed, this is just another way of expressing the notion of order-induction. The principle of infinite descent relies on the fact that well-orderings do not contain infinite descending chains to infer that a certain property does not hold for any element of a given set. The principle works as follows. Suppose we can show for every element x of the set A, if $P(x)$ holds, then there exists an element $y \in A$ with $y < x$ such that $P(y)$ holds. Now assume that there exists an element a such that $P(a)$ holds. Then we obtain an infinite descending chain of elements of A starting at a and satisfying the property P. But such an infinite descending chain does not exist. Hence, we conclude that no element of A satisfies P. We can express this principle as follows:

$$(\forall x \in A : P(x) \rightarrow (\exists y \in A : y < x \wedge P(y))) \rightarrow (\forall x \in A : \neg P(x)).$$

Depending on the formal system, the principle of infinite descent and the principle of induction may not be equivalent.

1.2 Inductive Theorem Proving

Inductive theorem proving is a branch of automated theorem proving which aims at automating the process of finding proofs for mathematical statements over some well-ordered structure such as for instance the natural numbers or any other inductively defined set. It is of particular interest to devise procedures which prove arithmetically true sentences. Because of Gödel's famous incompleteness theorems this problem is not only undecidable but it is not even recursively enumerable – that is, there is no sound procedure which effectively enumerates the arithmetically true sentences. Since it is impossible to prove exactly the arithmetically true sentences, it is necessary to

concentrate on less complex theories of arithmetic such as for instance Peano arithmetic and fragments thereof.

From a proof theoretical perspective the difficulty of inductive theorem proving manifests itself as the *non-analyticity* of induction invariants – that is, induction invariants are not always subformulas of the formula to be proved. Even though first-order logic is undecidable, effective proof search is possible because of the possibility to eliminate cuts. Informally, eliminating cuts means inlining the proofs of lemmas. A cut-free proof, that is to say a proof without explicit use of lemmas, is analytic in the sense that it only consists of subformulas of the formula to be proved. This restriction on the shape of the proofs makes it possible to search effectively for proofs. Since induction invariants are in general not analytic, inductive proofs do not satisfy the cut-elimination theorem. The cuts thus represent infinite branching points in the search space. Therefore effective proof search in this naive way is not feasible.

1.2.1 Approaches

The main problem of inductive theorem proving is thus to compute induction invariants which are sufficient to prove a given formula. A naive way to handle this problem is to prove formulas with respect to an a priori fixed set of induction axioms. But this is not a very flexible approach as it provides only a finite set of possible induction invariants. More sophisticated approaches try to discover induction invariants via cyclic proofs [BGP12, KP13], tree-grammars [EH15], rippling [BSVH$^+$93], etc. Sometimes these approaches were designed with respect to some precise notion of induction and sometimes they were not. In the latter case the set of sentences provable by the formalism at hand is not always apparent and needs further investigations.

1.2.2 Applications

The most notable application of inductive theorem proving is formal verification of software. Any reasonably complicated software involves loops or recursion or both. Correctness proofs need to be carried out in critical software for obvious reasons, but formal correctness proofs are also carried out to improve the overall quality of software. Proving correctness for programs involving non-trivial loops and/or recursion requires the use of inductive reasoning over inductive data types such as natural numbers, lists, trees, etc. Carrying out these correctness proofs manually is tedious, time-consuming and error-prone, hence automation is required. Another area of application of inductive theorem proving, and automated deduction in general, is mathematics itself. Automated theorem provers can assist mathematicians in formalizing proofs [BKPU16] and may even find proofs that are difficult for humans to find [McC97].

1.3 Problem Statement

We will now shortly discuss the problem that is approached in this thesis and how it will be approached.

1.3.1 The n-Clause Calculus

The *n-clause calculus* – a superposition calculus enhanced by a cycle detection mechanism originally introduced by Kersani and Peltier in [KP13] – is a formalism which was not designed to capture some a priori known type of induction. Until now it is not known how powerful the notion of cycles in this calculus actually is, but it is conjectured that it captures only a "weak" form of induction. Since this calculus reasons on natural numbers it seems reasonable to start the analysis of the provable sentences by describing them with respect to analogues of theories of arithmetic. This thesis aims at realizing the first step towards a better understanding of the n-clause calculus. More precisely, the aim of this thesis is the description of the sentences provable by the n-clause calculus with respect to the quantifier complexity of their induction invariants.

1.3.2 Approach

The problem of giving an upper bound on the quantifier complexity of the induction captured by the n-clause calculus will be approached by a two-step translation from the n-clause calculus to the calculus LKID using as intermediary representation the cyclic sequent calculus CLKID^ω.

The first step of the translation consists in translating the inductive cycles of the n-clause calculus to the cyclic sequent calculus CLKID^ω introduced by Brotherston and Simpson in [BS10]. This step of the translation mainly consists in providing a suitable representation for clause sets in terms of sequents. The inductive cycles will be converted into cycles of the calculus CLKID^ω. In this way we obtain an arguably more natural representation of the argument by infinite descent represented by inductive cycles. Since the notion of cycles of the calculus CLKID^ω is stronger than the notion of structural induction (see Section 4.4), this formalism seems to be a good starting point as a target formalism for the translation of inductive cycles whose strength is yet unknown.

The second step of the translation consists in translating a special type of cyclic proofs into inductive proofs. This step is somewhat critical since it has recently been shown that the notion of cycles is stronger than the notion of induction (see Section 4.4). Hence it is possible that the inductive cycles of the n-clause calculus cannot be simulated by structural induction LKID and thus that this step of the translation is not always possible. However it turned out that this is not the case, indeed the cyclic proofs obtained from the inductive cycles are simple enough that the translation to LKID is always possible.

1.4 Structure of the Thesis

This thesis is organized in several chapters. In Chapter 2 we introduce definitions and notations that we will need throughout this thesis. Chapter 3 introduces the n-clause calculus as well as its underlying logic – the n-clause logic. In Chapter 4 we present the first-order logic with inductive definitions as well as the calculi $\mathsf{LK}, \mathsf{LKID}, \mathsf{LKID}^\omega$, and CLKID^ω that serve as the target for the translation from the n-clause calculus.

Chapter 5 defines a translation from n-clause logic to first-order logic with inductive definitions and proves that this translation has some desirable properties. In Chapter 6 we will translate n-clause refutations to cyclic proofs. Moreover, we translate a specific type of cyclic proofs to inductive proofs, thereby obtaining the main theorem of this thesis stating that the n-clause calculus captures at most Σ_1-induction. In Chapter 7 we will discuss two natural questions that require further investigations. First we will discuss whether the derived induction invariants are optimal in the sense that the n-clause calculus indeed proves a sentence, which cannot be proven using only quantifier-free induction. Secondly, we will discuss the completeness of the n-clause calculus with respect to Σ_1-induction.

CHAPTER 2

Preliminary Definitions

This chapter introduces some notions related to terms that we will use throughout this thesis.

2.1 Term Languages

In the following we will define the notions of unsorted term languages and many-sorted term languages. Even though, unsorted term languages are in fact just a special case of many-sorted term languages, we define this notion separately in order to clearly distinguish between these two cases. This will be of particular relevance during the semantic analysis of the translation of clauses of the n-clause calculus to formulas of first-order logic with inductive definitions.

2.1.1 Unsorted Term Languages

Definition 2.1 (Variable Set)**.** *An untyped variable set $\mathcal{X}$ is a countable set of symbols.*

Definition 2.2 (Term Signature)**.** *A term signature Σ is a finite set of pairs of the form f/n, where f is called a function symbol and $n \in \mathbb{N}$ is the arity of f.*

Definition 2.3. *An untyped term language is a pair $(\Sigma, \mathcal{X})$, where Σ is a term signature, and $\mathcal{X}$ is a variable set such that $\{f : f/n \in \Sigma\} \cap \mathcal{X} = \varnothing$.*

Definition 2.4 (Untyped terms)**.** *Let $(\Sigma, \mathcal{X})$ be an untyped term language. Then the set of untyped terms $\mathcal{T}^{\Sigma}_{\mathcal{X}}$ is defined inductively as follows*

- $\mathcal{X} \subseteq \mathcal{T}^{\Sigma}_{\mathcal{X}}$,
- *if $f/n \in \Sigma$ and $t_1, \ldots, t_n \in \mathcal{T}^{\Sigma}_{\mathcal{X}}$, then $f(t_1, \ldots, t_n) \in \mathcal{T}^{\Sigma}_{\mathcal{X}}$.*

By $Gnd(\mathcal{T}^{\Sigma}_{\mathcal{X}})$ we denote the set of untyped ground terms.

2.1.2 Many-Sorted Term Languages

Definition 2.5 (Sort)**.** *A set of sorts is a finite set $\mathcal{S}$. The elements of $\mathcal{S}$ are called sorts.*

Definition 2.6 (Type)**.** *Let $\mathcal{S}$ be a set of sorts. A type τ is an expression of the form $s_1 \to \cdots \to s_n$ such that $s_1, \ldots, s_n \in \mathcal{S}$. We call the sort s_n the range of type τ.*

A type of the form $s_1 \to \cdots \to s_n \to s$ represents a function with domain $(s_1 \times \cdots \times s_n)$ and co-domain s.

Definition 2.7 (Many-Sorted Variable Set)**.** *Let $\mathcal{S} = \{s_1, \ldots, s_n\}$ be a set of sorts. A many-sorted variable set $\mathcal{X}$ over the sorts $\mathcal{S}$ assigns to each sort $s \in \mathcal{S}$ a variable set $\mathcal{X}_s$ such that $X_{s_1} \cap X_{s_2} = \varnothing$ for all $s_1, s_2 \in \mathcal{S}$ with $s_1 \neq s_2$.*

We write $x : s \in \mathcal{X}$ with to indicate that that $x \in \mathcal{X}_s$. Whenever we expect an untyped set of variables, then we may also use a many-sorted set of variables. Since the variable symbols are disjoint we interpret the many-sorted variable set $\mathcal{X} = (\mathcal{X}_1, \ldots, \mathcal{X}_n)$ as the untyped variable set $\bigcup_{s \in \mathcal{S}} \mathcal{X}_s$.

Definition 2.8 (Many-Sorted Term Signature)**.** *Let $\mathcal{S}$ be set of sorts. A many-sorted term signature Σ is a finite set of pairs of the form $f : \tau$, where τ is a type over $\mathcal{S}$. Moreover each function symbol f occurs at most once in Σ. Let $f : \tau$ be an element of Σ, then f is called a function symbol of type τ and the range of f is the range of τ.*

Definition 2.9 (Many-sorted Term Language)**.** *A many-sorted term language is a triple $(\mathcal{S}, \Sigma, \mathcal{X})$ where $\mathcal{S}$ is a set of sorts, Σ is a many-sorted term signature over $\mathcal{S}$ and $\mathcal{X}$ is a many-sorted variable set over $\mathcal{S}$ such that $\mathcal{X}_s \cap \{f \mid f : \tau \in \Sigma\} = \varnothing$ for all $s \in \mathcal{S}$*

Definition 2.10 (Many-Sorted Terms)**.** *Let $(\mathcal{S}, \Sigma, \mathcal{X})$ be a many-sorted term language. Then for each $s \in \mathcal{S}$ the set $\mathbb{T}^{\Sigma}_{\mathcal{X}}(s)$ of terms of sort s is defined inductively as follows*

- $\{x \mid x : s \in \mathcal{X}\} \subseteq \mathbb{T}^{\Sigma}_{\mathcal{X}}(s)$,
- *if $f : s_1 \to \cdots \to s_n \to s \in \Sigma$ and $t_i \in \mathbb{T}^{\Sigma}_{\mathcal{X}}(s_i)$ for all $i \in \{1, \ldots, n\}$, then $f(t_1, \ldots, t_n) \in \mathbb{T}^{\Sigma}_{\mathcal{X}}(s)$.*

By $Gnd(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s))$ we denote the set of ground s-terms.

Since we can reconstruct the respective arities of many-sorted function symbols, we will sometimes use many-sorted term languages at places where an untyped (i.e. one-sorted) term language is expected. For example the symbol $\mathsf{0} : \omega$ can be seen as a 0-ary function symbol i.e. as a constant symbol, and the symbol $\mathsf{s} : \omega \to \omega$ can be seen as a unary function symbol.

Example 1. Consider the following function symbols $\mathsf{0} : \omega, \mathsf{s} : \omega \to \omega, \mathsf{t} : \iota$ and $\mathsf{p} : \iota \to \omega \to \iota$ over the sorts ω and ι. The terms $\mathsf{s}(\mathsf{s}(\mathsf{0}))$, and $\mathsf{p}(\mathsf{p}(\mathsf{t}, \mathsf{0}), \mathsf{s}(\mathsf{0}))$ are well-typed, but the terms $\mathsf{s}(\mathsf{t})$ and $\mathsf{p}(\mathsf{0}, \mathsf{0})$ are not. The term $\mathsf{s}(\mathsf{t})$ is not well typed because t has type ι but s expected an ω-term as argument. The term $\mathsf{p}(\mathsf{0}, \mathsf{0})$ is not well-typed since it expects a ι-term as its first-argument but the ω-term $\mathsf{0}$ is given instead.

Definition 2.11. *Let $(\mathcal{S}, \Sigma, \mathcal{X})$ be a many-sorted term language. A substitution over this language is a function σ assigning to every variable $x : s \in \mathcal{X}$ a term $\sigma(x) \in \mathbb{T}^{\Sigma}_{\mathcal{X}}(s)$ such that $\{x \in \mathcal{X} \mid \sigma(x) \neq x\}$ is finite. The domain $dom(\sigma)$ is given by $dom(\Sigma) = \{x \in \mathcal{X} \mid \sigma(x) \neq s\}$.*

Consider a term language containing a sort ω and function symbols $\mathsf{0} : \omega$ and $\mathsf{s} : \omega \to \omega$, then for every $n \in \mathbb{N}$ we denote by $\overline{n}$ the term

$$\underbrace{\mathsf{s}(\mathsf{s}(\ldots \mathsf{s}}_{n \text{ times}}(\mathsf{0}) \ldots)).$$

We call this term the n-th numeral.

Definition 2.12. *Let $(\mathcal{S}, \Sigma, \mathcal{X})$ a be many-sorted term language. A congruence relation $\equiv$ is a function that assigns to each sort $s \in \mathcal{S}$ an equivalence relation $\equiv_s \subseteq Gnd(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s))^2$ such that*

$$\text{if } t_1 \equiv_{s_1} r_1, \ldots, t_n \equiv_{s_n} r_n \text{ then } f(t_1, \ldots, t_n) \equiv_s f(r_1, \ldots, r_n),$$

for all $t_i, r_i \in Gnd(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s_i))$, with $i \in \{1, \ldots, n\}$ and all $f : s_1 \to \ldots s_n \to s \in \Sigma$.

Definition 2.13. *Let A be an arbitrary set, $\equiv$ be an equivalence relation over A and let $a \in A$. Then we denote by $[a]_{\equiv} = \{b \in A : a \equiv b\}$ the equivalence class represented by element a.*

CHAPTER 3

Kersani and Peltier's calculus

In this section we will present the n-clause calculus. The n-clause calculus – originally introduced by Kersani and Pelter in [KP13] – is a superposition calculus enhanced with a cycle detection mechanism realizing a form of reasoning by infinite descent.

The superposition calculus is a refutational calculus. More precisely it is a refinement of the resolution calculus for equational logics, that is, logics whose formulas consist of equations only. Since clauses of such logics only consist of equations the resolution rule and the paramodulation rule can be combined into a single "superposition" rule.

In the context of the n-clause calculus the superposition calculus serves as the inference engine which means that it responsible for the generation of new clauses. The cycle detection mechanism will then try to detect an inductive dependency between subsets of clauses derived by the superposition calculus. Once such a cycle is detected the inductive information provided by this cycle allows the calculus to establish the unsatisfiability of the clause set at hand and therefore to terminate the refutation process.

3.1 The n-Clause Logic

The underlying logic of the n-clause calculus is the n-clause logic. This logic is a two-sorted[1] clausal, equational logic. One of its sorts represents natural numbers and the other sort represents some other data type for which arbitrary function symbols can be provided. In the following we will define the syntax and the semantics of the n-clause logic.

[1] We restrict ourselves to the case of a two-sorted logic because the article [KP13] restricts the logic to two sorts. There is however no apparent reason that this logic and the n-clause calculus cannot work in a many-sorted setting. Moreover the results described in Chapters 5 and 6 straightforwardly extend to the many-sorted case. Hence this restriction is more to be seen as a simplification.

3.1.1 Syntax

We start by defining the notion of signature of n-clause languages. As for first-order logic a signature is just a set of non-logical constants. In the case of n-clause logic a signature is actually just a set of function symbols. The n-clause logic provides the symbols $\mathsf{0}$, s, and η. The symbols $\mathsf{0}$ and s are function symbols of types ω and $\omega \to \omega$, respectively. The symbol η represents a fixed natural number. It can be seen as a Skolem constant obtained during the clausification phase by the Skolemization of an existential quantifier which does not occur in the scope of a universal quantifier. We assume that these symbols do not occur in n-clause signatures. We denote by Σ_N the term signature $\{\mathsf{0} : \omega, \mathsf{s} : \omega \to \omega\}$.

Definition 3.1 (Signature)**.** *An n-clause signature Σ is a two-sorted term signature over the sorts ω and ι such that $\Sigma_N \subseteq \Sigma$, no function symbol in Σ has range ω, and the symbol η does not occur in Σ.*

Definition 3.2 (Language)**.** *An n-clause language $L = (\mathcal{X}, \Sigma)$ is a two-sorted term language over sorts ι and ω, where Σ is an n-clause signature and $\eta \notin \mathcal{X}$.*

As the name of the logic suggests the main syntactic construct of the n-clause logic are so-called *n-clauses* – sometimes also called *constraint clauses*. An n-clause is very similar to a usual clause but it also contains constraint atoms of the form $\eta \simeq t$, where t is an ω-term. An n-clause is to be interpreted as an implication where the constraint atoms are in the antecedent and the other literals are in the succedent.

Definition 3.3 (Literals, Clauses, n-Clauses)**.** *Let $L = (\mathcal{X}, \Sigma)$ be an n-clause language, then we define the notions of literals, clauses, constraint clauses and clause sets as follows:*

- *If $t_1, t_2 \in \mathbb{T}^{\Sigma}_{\mathcal{X}}(\iota)$ and $\bowtie \in \{\simeq, \not\simeq\}$, then $t_1 \bowtie t_2$ is a literal*
- *A clause is a finite set of literals,*
- *If C is a clause and $t_1, \ldots, t_n \in \mathbb{T}^{\Sigma}_{\mathcal{X}}(\omega)$, then $[C \mid \eta \simeq t_1, \ldots, \eta \simeq t_n]$ is a constraint clause.*
- *A clause set is a finite set of constraint clauses.*

For a constraint clause $\mathcal{C} = [C \mid \eta \simeq t_1, \ldots, \eta \simeq t_n]$ we define $ctr(\mathcal{C}) = \{\eta \simeq t_1, \ldots, \eta \simeq t_n\}$ and $cls(\mathcal{C}) = C$.

We denote by $\square$ an empty clause part and $\Diamond$ denotes an empty constraint. Whenever a concrete constraint clause has an empty constraint part, then we omit the $\Diamond$ and the brackets, that is we write this constraint clause as a usual clause. Since the clause part is interpreted disjunctively, we will sometimes use the symbol $\vee$ to indicate set-theoretic union on clause parts. Similarly, we use the symbol $\wedge$ for the union of constraint parts.

Example 2. We consider a language containing the symbols $\mathsf{t} : \iota$, $\mathsf{g} : \iota \to \iota$ and $\mathsf{p} : \omega \to \iota \to \iota$. The following are constraint clauses.

$$\mathsf{p}(\mathsf{0}, \mathsf{t}) \simeq \mathsf{t} \tag{E1}$$

$$\mathsf{p}(x, y) \not\simeq \mathsf{t} \vee \mathsf{p}(\mathsf{s}x, \mathsf{g}y) \simeq \mathsf{t} \tag{E2}$$

$$[\mathsf{p}(x, y) \not\simeq \mathsf{t} \mid \eta \simeq x] \tag{E3}$$

Intuitively, the constant symbol t represents the truth value true, p represents a predicate, and g represents some function. Therefore, clause (E1) indicates that $\mathsf{p}(\mathsf{0}, \mathsf{t})$ is true, clause (E2) indicates that $\mathsf{p}(x, y)$ implies $p(\mathsf{s}x, \mathsf{g}y)$, and clause (E3) is intended to express that for every x equal to η the formula $\mathsf{p}(x, y)$ is not true for any y.

3.1.2 Semantics

The syntax being defined we will now define the semantics of the n-clause logic. Since the n-clause logic is an equational logic, its semantics are most naturally defined in terms of congruence relations over the ground terms. Variables will be interpreted as ranging over the set of ground terms of their respective types.

Definition 3.4 (Interpretation)**.** *Let $L = (\mathcal{X}, \Sigma)$ be an n-clause language. An L-interpretation is a pair $\mathcal{I} = (n^{\mathcal{I}}, \equiv^{\mathcal{I}})$ where $n^{\mathcal{I}} \in \mathbb{N}$ and $\equiv^{\mathcal{I}}$ is a congruence relation over L and $\equiv^{\mathcal{I}}_{\omega}$ is the syntactic equality. We define $\eta^{\mathcal{I}} = n^{\mathcal{I}}$.*

The notion of truth under an interpretation for ground clauses is straightforward. Non-ground clauses are considered as universally quantified. Constraint clauses behave as implications whose antecedent is formed by the conjunction of the atoms $\eta \simeq t$ of the constraint part.

Definition 3.5. *Let $L = (\mathcal{X}, \Sigma)$ be an n-clause language and $\mathcal{I}$ an L-interpretation. The notion of truth is defined as follows:*

- *Let $t_1 \simeq t_2$ be a ground literal. We define $\mathcal{I} \models t_1 \simeq t_2$ if $t_1 \equiv^{\mathcal{I}} t_2$;*
- *Let $t_1 \not\simeq t_2$ be a ground literal. We define $\mathcal{I} \models t_1 \not\simeq t_2$ if $\mathcal{I} \not\models t_1 \simeq t_2$;*
- *Let C be a ground clause. We define $\mathcal{I} \models C$ if there exists a literal $l \in C$ such that $\mathcal{I} \models l$;*
- *Let $\mathcal{C} = [C \mid \eta \simeq t_1, \ldots, \eta \simeq t_n]$ be a constraint clause. We define $\mathcal{I} \models \mathcal{C}$ if every ground substitution σ with $dom(\sigma) = var(\mathcal{C})$ and $\overline{\eta^{\mathcal{I}}} = t_i$ for all $i = 1, \ldots, n$ satisfies $\mathcal{I} \models C\sigma$.*
- *Let S be a clause set, then $\mathcal{I} \models S$ if $\mathcal{I} \models \mathcal{C}$ for every $\mathcal{C} \in S$.*

Let S be a clause set and I an interpretation. We say that $\mathcal{I}$ *is a model of* S – or, S *is true under* $\mathcal{I}$ – if $\mathcal{I} \models S$. The notions of satisfiability, validity and entailment are defined in analogy to first-order logic.

Definition 3.6. *Let L be an n-clause language and S an L-clause set, then S is called satisfiable if there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models S$. We call S valid if S is true under every interpretation $\mathcal{I}$. Let S_1, S_2 be L-clause sets, then we define $S_1 \models S_2$ to be true if every model of S_1 is a model of S_2.*

Consider again the clauses (E1), (E2) and (E3) of Example 2. These constraint clauses are unsatisfiable with respect to the semantics defined above. Let $\mathcal{I} = (m, \equiv)$ be an arbitrary model of the these three clauses. We claim that $\mathcal{I} \models \mathsf{p}(\overline{n}, \mathsf{g}^n(\mathsf{t})) \simeq \mathsf{t}$ for all $n \in \mathbb{N}$. For the base case observe that $\mathcal{I} \models \mathsf{p}(0, \mathsf{t}) \simeq \mathsf{t}$ by the assumption that $\mathcal{I}$ is a model of the clause (E1). For the induction step we suppose that $n > 0$. Since $\mathcal{I}$ is a model of the clause (E2) we have by the definition of the semantics $\mathcal{I} \models \mathsf{p}(\overline{n-1}, g^{n-1}\mathsf{t}) \not\simeq \mathsf{t} \vee \mathsf{p}(\overline{n}, \mathsf{g}^n\mathsf{t}) \simeq \mathsf{t}$. By the induction hypothesis we moreover have $\mathcal{I} \models \mathsf{p}(\overline{n-1}, g^{n-1}\mathsf{t}) \simeq \mathsf{t}$. Hence again by the semantics of the logic it must be the case that $\mathcal{I} \models \mathsf{p}(\overline{n}, \mathsf{g}^n\mathsf{t}) \simeq \mathsf{t}$. Now consider the ground substitution

$$\sigma = \{x \mapsto \overline{m}, y \mapsto \mathsf{g}^m\mathsf{t}\},$$

and observe that $x\sigma = \overline{m} = \overline{\eta^{\mathcal{I}}}$ but $\mathcal{I} \not\models \mathsf{p}(\overline{m}, \mathsf{g}^m\mathsf{t}) \not\simeq \mathsf{t}$ which means that $\mathcal{I}$ is not a model of clause (E3). Contradiction! Observe that we have shown unsatisfiability of this clause set via a quantifier-free weak induction. This was possible because we are able to express the iteration of the function g. As we shall later see in Section 7.1, it is questionable whether this clause set is refutable by a quantifier-free induction in a formal system with a restricted language.

Constraint clauses can be normalized in the sense that every constraint clause having a non-empty constraint part is logically equivalent to a constraint clause having exactly one atom in its constraint part.

Definition 3.7. *A constraint clause $\mathcal{C}$ is normal if it is either of the form $\mathcal{C} = [C \mid \eta \simeq t]$, or of the form $\mathcal{C} = [C \mid \Diamond]$.*

Proposition 3.1. *Let $\mathcal{C} = [C \mid \eta \simeq t_1, \ldots, \eta \simeq t_n]$ be a constraint clause. If $t_1, \ldots, t_n$ are unifiable with m.g.u. σ, then $\mathcal{C}$ is logically equivalent to the clause constraint clause $[C\sigma \mid \eta \simeq t_1\sigma]$. Otherwise, $\mathcal{C}$ is valid i.e. $\mathcal{C}$ is logically equivalent to the clause $[y \simeq y \mid \eta \simeq 0]$.*

Proof. Let us start by assuming that the terms $t_1, \ldots, t_n$ are not unifiable. Let $\mathcal{I}$ be any interpretation and σ be a ground substitution with $dom(\sigma) = var(\mathcal{C})$. Since $t_1, \ldots, t_n$ are not unifiable, σ is not a unifier. Thus, there exist $i, j \in \{1, \ldots, n\}$ such that $t_i\sigma \neq t_j\sigma$. Hence, we either have $\overline{\eta^{\mathcal{I}}} \neq t_i$ or $\overline{\eta^{\mathcal{I}}} \neq t_j$. Therefore $\mathcal{I} \models \mathcal{C}$.

Let us now assume that $t_1, \ldots, t_n$ are unifiable. We need to show that $\mathcal{I} \models \mathcal{C}$ if and only if $\mathcal{I} \models [C\sigma \mid \eta \simeq t_1\sigma]$. Let $\mathcal{I}$ be any interpretation. For the "only if" direction we proceed indirectly and assume that $\mathcal{I} \models \mathcal{C}$, but $\mathcal{I} \not\models [C\sigma \mid \eta \simeq t_1\sigma]$. Then there exists a ground substitution σ' with domain $var([C\sigma \mid \eta \simeq t_1\sigma])$ such that $\overline{\eta^{\mathcal{I}}} = t_1\sigma\sigma'$ and $\mathcal{I} \not\models C\sigma\sigma'$. Since σ is the m.g.u. of the terms $t_1, \ldots, t_n$, the substitution $\sigma\sigma'$ is also a unifier of these terms. Moreover, $\sigma\sigma'$ clearly is a ground substitution with domain

$var(\mathcal{C})$. Hence we have $\overline{\eta^{\mathcal{I}}} = t_i\sigma\sigma'$ for all $i \in \{1, \ldots, n\}$, but we also have $\mathcal{I} \not\models C\sigma\sigma'$. Hence $\mathcal{I} \not\models \mathcal{C}$. Contradiction!

For the "if" direction let us assume that $\mathcal{I} \models [C\sigma \mid \eta \simeq t_1\sigma]$ and $\mathcal{I} \not\models \mathcal{C}$. From this it follows that there exists a ground substitution σ' such that $\overline{\eta^{\mathcal{I}}} = t_i\sigma'$, for all $i \in \{1, \ldots, n\}$. In other words σ' is a unifier of the terms $t_1, \ldots, t_n$. Since σ is an m.g.u. for these terms, there exists a substitution θ such that $\sigma' = \sigma\theta$. Hence $t_1\sigma\theta = t_1\sigma'$ and moreover $C\sigma\theta = C\sigma'$. Therefore, $\mathcal{I} \not\models [C\sigma \mid \eta \simeq t_1\sigma]$. This contradicts the assumptions. □

Definition 3.8. *Let $\mathcal{C} = [C \mid X]$ be a constraint clause. Then the function $\|\cdot\|$ is defined by*

$$\|C\| = \begin{cases} \mathcal{C} & \text{if } |X| \leq 1 \\ [C\sigma \mid \eta \simeq t\sigma] & \text{if } |X| > 2,\ t \in X, \text{ and } X \text{ is unifiable with m.g.u. } \sigma \\ [y \simeq y \mid \eta \simeq 0] & \text{otherwise.} \end{cases}$$

3.2 The n-Clause Calculus

The n-clause calculus is a refutational calculus for n-clause sets, that is it shows the unsatisfiability of clause sets. It consists of a superposition calculus which plays to role of inference engine and a mechanism for the detection of inductive dependencies between derived n-clauses. We will first present the superposition calculus as described in [KP13] and its related notions. Secondly we shall see the semantic notion of inductive loops and finally a syntactic variant called inductive cycles.

3.2.1 The Superposition Calculus

The first-order inference rules of this superposition calculus are the rules of superposition, reflection and factorization. Before we introduce these inference rules we will have to introduce the notions of reduction orderings and selection functions, which guide the derivation of new clauses. For the sake of simplicity we define these concepts on untyped term languages, however, these concepts naturally generalize to the case of typed term languages.

The position relation defined below allows us to designate a subterm occurrence in a given term. Subterm positions are expressed as paths in the tree representation of terms. The position of a subterm is a list of integers that represent the path from the root node to the head symbol of the subterm. The integers in this list specify the branch to be taken at each step. This is required to specify the term replacement carried out by the superposition rule. We represent the empty sequence by ε and concatenation by $\cdot$.

Definition 3.9 (Subterm positions)**.** *Let $(\Sigma, \mathcal{X})$ be a term language. The set of positions* $\mathrm{Pos}(t)$ *of a term $t \in \mathcal{T}^{\Sigma}_{\mathcal{X}}$ is defined inductively by*

- *If $t \in \mathcal{X}$, then* $\mathrm{Pos}(t) = \{\varepsilon\}$,

- *If t is of the form $f(t_1, \ldots, t_n)$, then*

$$\mathrm{Pos}(t) = \{\varepsilon\} \cup \bigcup_{i=1}^{n} \{i \cdot p : p \in \mathrm{Pos}(t_i)\}.$$

The term positions can then be used to indicate subterm occurrences in the way described above and it can be used to specify term replacements – that is a subterm occurrence at a given position is to be replaced by another term.

Definition 3.10 (Term index)**.** *Let $(\Sigma, \mathcal{X})$ be a term language. For a term $t \in \mathcal{T}_{\mathcal{X}}^{\Sigma}$ we define*

$$t|_{\varepsilon} = t.$$

For a term $t \in \mathcal{T}_{\mathcal{X}}^{\Sigma}$ of the form $f(t_1, \ldots, t_n)$ and a position $i \cdot p \in \mathrm{Pos}(t)$ with $1 \leq i \leq n$ we define

$$t|_{i \cdot p} = t_i|_p.$$

Definition 3.11 (Term replacement)**.** *Let $(\Sigma, \mathcal{X})$ be a term language. For terms $s, t \in \mathcal{T}_{\mathcal{X}}^{\Sigma}$ we define*

$$s[t]_{\varepsilon} = t.$$

For $s \in \mathcal{T}_{\mathcal{X}}^{\Sigma}$ of the form $f(u_1, \ldots, u_n)$, $t \in \mathcal{T}_{\mathcal{X}}^{\Sigma}$ and $i \cdot p \in \mathrm{Pos}(s)$ with $1 \leq i \leq n$ we define

$$s[t]_{i \cdot p} = f(u_1, \ldots, u_{i-1}, u_i[p], u_{i+1}, \ldots, u_n).$$

We will now define the notion of reduction ordering and selection function. These two notions are used by the superposition calculus to select clause and literals for the application of inference rules.

Definition 3.12. *Let $(\Sigma, \mathcal{X})$ be a term language. We say that a binary relation $\succ$ on $\mathcal{T}_{\mathcal{X}}^{\Sigma}$ is compatible with Σ-operations if*

$$t_i \succ t_i' \text{ implies } f(t_1, \ldots, t_{i-1}, t_i, t_{i+1}, \ldots, t_n) \succ f(t_1, \ldots, t_{i-1}, t_i', t_{i+1}, \ldots, t_n)$$

for all $f/n \in \Sigma$, all $t_1, \ldots, t_n, t_i' \in \mathcal{T}_{\mathcal{X}}^{\Sigma}$, and all $i \in \{1, \ldots, n\}$.

Definition 3.13. *Let $(\Sigma, \mathcal{X})$ be a term language. A binary relation $\succ$ over $\mathcal{T}_{\mathcal{X}}^{\Sigma}$ is called stable under substitutions, if $s \succ s'$ implies $s\sigma \succ s'\sigma$ for all $s, s' \in \mathcal{T}_{\mathcal{X}}^{\Sigma}$ and substitutions σ.*

Definition 3.14. *Let $(\Sigma, \mathcal{X})$ be a term language. A binary relation $\succ$ is called a rewrite relation, if it is compatible with Σ-operations and stable under substitutions. A strict partial ordering over $\mathcal{T}_{\mathcal{X}}^{\Sigma}$ that is a rewrite relation is called rewrite ordering.*

Definition 3.15 (Reduction Ordering)**.** *A well-founded rewrite ordering is called reduction ordering.*

Definition 3.16 (Selection Function)**.** *Let $\succ$ be a reduction ordering, then a function $sel_\succ$ is called a selection function if for a given clause C it holds $sel_\succ(C) \subseteq C$ and either $sel_\succ(C)$ contains a negative literal or $sel_\succ(C)$ contains the maximal literals of C with respect to $\succ$.*

Definition 3.17. *A superposition system is a pair $\mathfrak{S} = (\succ, sel_\succ)$ where $\succ$ is a reduction ordering and $sel_\succ$ is a selection function.*

The superposition rule resembles the resolution rule in the way it combines two clauses, it is also similar to the paramodulation inference since it also realizes a term replacement at an arbitrary depth. Note that the inference rules described below use normal clauses as premises. This is according to the calculus defined in the article [KP13]. According to Kersani and Peltier, normalization in this superposition calculus is applied in a systematic way to every generated n-clause. This is perhaps not the best way to describe normalization. From a theoretical perspective, a more natural way to handle normalization, would be to introduce separate inference rules. In addition this would be more convenient when dealing with normal forms of derivations.

Definition 3.18 (Superposition)**.** *Let L be an n-clause language and $\mathfrak{S} = (\succ, sel_\succ)$ a superposition system. Furthermore let $\mathcal{C} = [C \vee t \bowtie s \mid X]$ and $\mathcal{D} = [D \vee u \simeq v \mid Y]$ be normal L-clauses such $\bowtie \in \{\simeq, \not\simeq\}, \sigma = mgu(u, t|_p), v\sigma \not\succ u\sigma, s\sigma \not\succ t\sigma, t|_p$ is not a variable and $(t \bowtie s)\sigma \in sel_\succ((C \vee t \bowtie s)\sigma), (u \simeq v)\sigma \in sel_\succ((D \vee u \simeq v)\sigma)$. Then we define*

$$\sup^{\mathfrak{S}}_{t \bowtie s, u \simeq v}(\mathcal{C}, \mathcal{D}) = [C \vee D \vee t[v]_p \bowtie s \mid X \wedge Y]\sigma.$$

The reflection inference rule eliminates a negative literal that becomes redundant under a suitable substitution.

Definition 3.19. *Let L be an n-clause language and $\mathfrak{S} = (\succ, sel_\succ)$ a superposition system. Let $\mathcal{C} = [C \vee t \not\simeq s \mid X]$ be a normal L-clause such that $\sigma = mgu(t, s)$, $(t \not\simeq s)\sigma \in sel_\succ((C \vee t \not\simeq s)\sigma)$, then we define*

$$\mathrm{refl}^{\mathfrak{S}}_{t \not\simeq s}(\mathcal{C}) = [C \mid X]\sigma.$$

Factorization is used to reformulate two redundant positive literals as a negative literal and a positive literal.

Definition 3.20. *Let L be an n-clause language and $\mathfrak{S} = (\succ, sel_\succ)$ a superposition system. Let $\mathcal{C} = [C \vee t \simeq s \vee u \simeq v \mid X]$ be a normal L clause with $\sigma = mgu(t, u), s\sigma \not\succ t\sigma, v\sigma \not\succ u\sigma, (t \simeq s)\sigma \in sel_\succ((C \vee t \simeq s \vee u \simeq v)\sigma)$, then we define*

$$\mathrm{fact}^{\mathfrak{S}}_{t \simeq s, u \simeq v}(\mathcal{C}) = [C \vee s \not\simeq v \vee t \simeq s \mid X]\sigma$$

Whenever the superposition system and/or the literals that are being used for an inference are irrelevant, then we will drop the superscript and the subscript. New clause sets can be derived by repeated application of the three inference rules presented above.

We will represent deductions in linear form but it is clearly also possible to represent these deductions as trees. The tree representation needs nodes to be duplicated in each level and is thus in general exponential in the size of the linear representation.

Observe that the superposition inference (Definition 3.18) increases in general the size of the constraint part of the resulting clause. In order to obtain useful clauses for the detection of inductive cycles, we shall assume – as in [KP13] – that clauses are normalized in a systematic way.

Definition 3.21. *Let L be an n-clause language, $\mathfrak{S}$ a superposition system, and S an L-clause set. A $\mathfrak{S}$-superposition deduction from S is a sequence of L-clauses $\mathcal{C}_1, \ldots, \mathcal{C}_n$ such that for every $1 \leq i \leq n$ one of the following holds.*

1. *$\mathcal{C}_i \in S$,*
2. *there exist $j, k < i$, such that $\mathcal{C}_i = \sup^{\mathfrak{S}}(\mathcal{C}'_j, \mathcal{C}'_k)$, where $\mathcal{C}'_j$, $\mathcal{C}'_k$ are variable-disjoint copies of $\mathcal{C}_j$ and $\mathcal{C}_k$, respectively,*
3. *there exists $j < i$ such that $\mathcal{C}_i = \mathrm{refl}^{\mathfrak{S}}(\mathcal{C}_k)$,*
4. *there exists $j < i$ such that $\mathcal{C}_i = \mathrm{fact}^{\mathfrak{S}}(\mathcal{C}_j)$,*
5. *there exists $j < i$ such that $\mathcal{C}_i = \|\mathcal{C}_j\|$.*

A superposition deduction $\mathcal{C}_1, \ldots, \mathcal{C}_n$ with $\mathcal{C}_n = [\Box \mid \Diamond]$ is called a superposition refutation.

We write $S \vdash \mathcal{C}$ to indicate that there exists a superposition deduction of the clause $\mathcal{C}$ from the clause set S. Similarly, we write $S_1 \vdash S_2$ to indicate that $S_1 \vdash \mathcal{C}$ for all $\mathcal{C} \in S_2$. We write $\pi : S_1 \vdash S_2$ to denote that π is a superposition deduction of S_2 from S_1.

For an $n \in \mathbb{N}$ we denote by $\eta \not\approx n$, the clause $[\Box \mid \eta \simeq \overline{n}]$. Let us again consider the clauses (E1), (E2) and (E3) of Example 2 and observe which clauses are derivable from these clauses. Using the superposition calculus we can generate all clauses of the form $\eta \not\approx n$ with $n \in \mathbb{N}$. We start by deriving the clause $\mathsf{p}(\overline{1}, \mathsf{g}^1\mathsf{t}) \simeq \mathsf{t}$ as follows:

$$\frac{\mathsf{p}(0,\mathsf{t}) \simeq \mathsf{t} \qquad \mathsf{p}(x,y) \not\simeq \mathsf{t} \vee \mathsf{p}(sx, \mathsf{g}y) \simeq \mathsf{t}}{\mathsf{p}(\overline{1}, \mathsf{g}^1\mathsf{t}) \simeq \mathsf{t}.}\ \sup$$

We can now superpose (E2) and $\mathsf{p}(\overline{1}, \mathsf{g}^1\mathsf{t}) \simeq \mathsf{t}$ to obtain the clause $\mathsf{p}(\overline{2}, \mathsf{g}^2\mathsf{t}) \simeq \mathsf{t}$. By iterating this procedure we derive all clauses of the form $\mathsf{p}(\overline{n}, \mathsf{g}^n\mathsf{t}) \simeq \mathsf{t}$ with $n \in \mathbb{N}$. By superposing each of these clause with clause (E3) we finally obtain the clauses $\eta \not\approx n$ for all $n \in \mathbb{N}$. But we are apparently not able to derive a contradiction and thus are unable to obtain a refutation. In order to refute clause sets such as the one in Example 2, we need to enhance the superposition calculus by some mechanism that captures some adequate inductive information.

3.2.2 Cycle Detection

We will now introduce inductive cycles and its related notions. Inductive cycles are the core of the inductive reasoning captured by the n-clause calculus. Informally, an inductive cycle is a clause set $S(\eta)$ that behaves as an inductive formula but instead of inheriting truth towards larger numbers it propagates truth towards smaller numbers, thereby realizing a form of argumentation by infinite descent. In other words $S(\eta)$ implies $S(\eta - j)$ for some natural number $j > 0$, and $S(0), \ldots, S(j-1)$ are unsatisfiable.

In order to realize such arguments by infinite descent on η, we need a way to express subtraction of j from η. Since there is no predecessor in the n-clause logic, we need to encode subtraction in a different way.

For simplicity let us first consider the clause $\mathcal{C}(\eta) = [C(x) \mid \eta \simeq x]$. We want to express the subtraction of one from η. The parameter η is represented in $C(x)$ by the variable x. Therefore, instead of decreasing η by one we can as well decrease the value of x by one. We can accomplish this by setting $\mathcal{C}(\eta - 1) = [\underline{C(x)} \mid \eta \simeq \mathsf{s}x]$. For a given interpretation $\mathcal{I}$ the variable x will always be interpreted as $\eta^{\mathcal{I}} - 1$. In a general setting the subtraction is realized by the operation $\downarrow$ defined below.

Definition 3.22. *Let L be an n-clause language, $\mathcal{C} = [C \mid \eta \simeq t_1, \ldots, \eta \simeq t_n]$ an L-constraint clause and $j \in \mathbb{N}$. Then the L-constraint clause $\mathcal{C}{\downarrow_j}$ is given by:*

$$[C \mid \eta \simeq \mathsf{s}^j t_1, \ldots, \eta \simeq \mathsf{s}^j t_n].$$

For a L-clause set S we define $S{\downarrow_j} = \{\mathcal{C}{\downarrow_j} \mid \mathcal{C} \in S\}$.

Note that clauses with an empty constraint are clearly not affected by this operation. Informally, the following proposition states that the operation $\downarrow$ captures subtraction by j.

Proposition 3.2. *Let L be an n-clause language, S an L-clause set, $i, j \in \mathbb{N}$, and $\mathcal{I}$ an interpretation with $\eta^{\mathcal{I}} = i + j$. If $\mathcal{I} \models S{\downarrow_j}$, then $\mathcal{J} = \mathcal{I} \cup \{\eta \to i\}$ satisfies $\mathcal{J} \models S$.*

Proof. See proof of Proposition 3 in [KP13]. □

Inductive Loops Let us first define the concept of inductive loop. The concept of inductive loop is a semantic notion of inductivity for clause sets. But it is not of practical value since the entailment between clause sets is undecidable. However, inductive loops will serve us to understand on the metalevel the notion of induction captured by the syntactic inductive cycles.

Definition 3.23. *Let L be an n-clause language and let S be an L-clause set. Then a triple (i, j, S') with $j > 0$ and $S' \subseteq S$ is an inductive loop for S if $S' \models \eta \not\simeq i + k$ for all $0 \leq k < j$ and $S' \models S'{\downarrow_j}$.*

In order to clarify the notion of inductive loops let us look at an example.

Example 3. Consider again the clauses (E1), (E2) and (E3) of Example 2. We have earlier shown that clauses (E1) and (E3) derive the clause $\eta \not\approx 0$ via a single superposition inference. Moreover, we can derive the clause $[\mathsf{p}(x,y) \not\simeq \mathsf{t} \mid \eta \simeq \mathsf{s}x]$ from (E2) and (E3) as follows:

$$\dfrac{\dfrac{\mathsf{p}(x,y) \not\simeq \mathsf{t} \vee \mathsf{p}(\mathsf{s}x,\mathsf{g}y) \simeq \mathsf{t} \qquad [\mathsf{p}(u,v) \not\simeq \mathsf{t} \mid \eta \simeq u]}{[\mathsf{p}(x,y) \not\simeq \mathsf{t} \vee \mathsf{t} \not\simeq \mathsf{t} \mid \eta \simeq \mathsf{s}x]}\,\mathrm{sup}_{\mathsf{p}(\mathsf{s}x,\mathsf{g}y)\simeq\mathsf{t},\mathsf{p}(u,v)\not\simeq\mathsf{t}}}{[\mathsf{p}(x,y) \not\simeq \mathsf{t} \mid \eta \simeq \mathsf{s}x].}\,\mathrm{refl}_{\mathsf{t}\not\simeq\mathsf{t}}$$

In other words clauses (E2) and (E3) entail $(\text{E3})\!\downarrow_1$. Since clauses (E1) and (E2) have an empty constraint they are invariable with respect to the operation $\downarrow$. Hence the triple $(0, 1, \{(\text{E1}), (\text{E2}), (\text{E3})\})$ is an inductive loop.

We will now describe the properties of inductive loops. This will help us understand what kind of information we can obtain from the existence of an inductive loop.

Proposition 3.3. *Let L be an n-clause language, S an L-clause set and (i, j, S') an inductive loop for S. Then we have $S \models \eta \not\approx n$ for all $n \geq i$.*

Proof. Let $S' \subseteq S$ such that $S' \models [\Box \mid \eta \simeq \mathsf{s}^k 0]$ for all $i \leq k < i + j$ and $S' \models S'\!\downarrow_j$. We proceed by induction on the natural number n.

For the base cases let $i \leq n < i + j$ and consider an arbitrary interpretation $\mathcal{I}$ such that $I \models S$. Since $S' \subseteq S$ and $S' \models [\Box \mid \eta \simeq \mathsf{s}^k 0]$ for all $i \leq k < i + j$ we have in particular $I \models [\Box \mid \eta \simeq \mathsf{s}^n 0]$.

For the induction step suppose that $S \models [\Box \mid \eta \simeq \mathsf{s}^n 0]$ and suppose that $S \not\models [\Box \mid \eta \simeq \mathsf{s}^{n+j} 0]$. Then there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models S$ but $\eta^{\mathcal{I}} = n+j$. Then since $S' \subseteq S$ we also have $\mathcal{I} \models S\!\downarrow_j$ and by Proposition 3.2 we obtain that $\mathcal{J} = \mathcal{I} \cup \{\eta \mapsto n\}$ satisfies $\mathcal{J} \models S \models [\Box \mid \eta \simeq \mathsf{s}^n]$. But $\overline{\eta^{\mathcal{J}}} = \mathsf{s}^n 0$ and $\mathcal{J} \not\models \Box$; hence $\mathcal{J} \not\models [\Box \mid \eta \simeq \mathsf{s}^n 0]$. This contradicts the assumptions. We conclude that $S \models [\Box \mid \eta \simeq \mathsf{s}^n 0]$ for all $n \geq i$. □

The following theorem justifies the correctness of the inductive cycles that are going to be defined. Informally, the clause $[\Box \mid \eta \simeq \mathsf{s}^i x]$ with $i \in \mathbb{N}$ expresses that η is strictly less than i. Therefore we will abbreviate this clause by $\eta \prec i$. Remember that we abbreviate $[\Box \mid \eta \simeq k]$ by $\eta \not\approx k$. These abbreviations will allow use to state the following definitions and results in a much more comprehensible way.

Theorem 3.4. *Let L be an n-clause language and S an L-clause set such that S admits an inductive loop (i, j, S'), then $S \models \eta \prec i$.*

Proof. We proceed indirectly and assume that there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models S$ and $\mathcal{I} \not\models [\Box \mid \eta \simeq \mathsf{s}^i x]$ i.e. there exists a typed ground substitution σ such that $\overline{\eta^{\mathcal{I}}} = \mathsf{s}^i(x\sigma)$. Since $x\sigma$ is ground there exists $k \in \mathbb{N}$ such that $\eta^{\mathcal{I}} = \overline{i+k}$. This is impossible because of Proposition 3.3. □

Inductive Cycles We are now going to introduce the notion of inductive cycle. Inductive loops are by their semantic nature not suitable for formal reasoning, hence we need to restrict this notion in order to obtain a practically useful calculus. Inductive cycles are obtained by restricting the semantic entailment to the derivability with respect to the underlying superposition calculus, but inductive cycles include some further restrictions. We will discuss this in the following.

Definition 3.24. *Let L be an n-clause language, S a set of L-clauses and $\mathfrak{S}$ a superposition system. Then an inference relation δ for S is a partial function $\delta : S \to \mathcal{P}(S)$ such that for every $\mathcal{C} \in S$ whenever $\delta(\mathcal{C})$ is defined, then one of the following holds:*

- *there exists $\mathcal{D}_1, \mathcal{D}_2 \in \delta(\mathcal{C})$ such that $\sup^{\mathfrak{S}}(\mathcal{D}_1, \mathcal{D}_2) = \mathcal{C}$.*
- *there exists $\mathcal{D} \in \delta(\mathcal{C})$ such that $\mathrm{refl}^{\mathfrak{S}}(\mathcal{D}) = \mathcal{C}$.*
- *there exists $\mathcal{D} \in \delta(\mathcal{C})$ such that $\mathrm{fact}^{\mathfrak{S}}(\mathcal{D}) = \mathcal{C}$.*

In other words an inference relation δ is just a restriction of the inference relation induced by the superposition calculus.

Definition 3.25. *Let L be an n-clause language, S a set of L-clauses, $\mathfrak{S}$ a superposition system and δ an inference relation for S. Then the relation $\vdash_\delta \subseteq \mathcal{P}(S)^2$ is the smallest relation such that for all $S_1, S_2 \subseteq S$ we have $S_1 \vdash_\delta S_2$ if for every $\mathcal{C}_2 \in S_2$ one of the following holds:*

- *$\mathcal{C}_2 \in S_1$,*
- *$\delta(\mathcal{C}_2)$ is defined and $S_1 \vdash_\delta \delta(\mathcal{C}_2)$,*
- *$ctr(\mathcal{C}_2) = \Diamond$.*

For a given set of clauses S we will denote by $S[\top]$ the set $\{\mathcal{C} \in S \mid ctr(\mathcal{C}) = \Diamond\}$.

Lemma 3.5. *Let L be an n-clause language, S an L-clause set and δ an inference relation for S. Then for every $S_1, S_2 \subseteq S$, if $S_1 \vdash_\delta S_2$, then $S_1 \cup S[\top] \models S_2 \cup S[\top]$.*

Proof. The article [KP13] does not mention a proof. We obtain a proof via Propositions 5.11 and 6.13. This proof gives a more restricted view of the semantic entailment. □

Definition 3.26. *Let L be an n-clause language. An immediate entailment relation is a decidable relation $\sqsupseteq$ between L-clauses such that $\mathcal{C} \sqsupseteq \mathcal{D}$ implies $\mathcal{C} \vdash \mathcal{D}$. The relation $\sqsupseteq$ is extended to L-clause sets as follows: Let S_1, S_2 L-clause sets, then $S_1 \sqsupseteq S_2$ if and only if for every $\mathcal{C}_2 \in S_2$ there exists a $\mathcal{C}_1 \in S_1$ such that $\mathcal{C}_1 \sqsupseteq \mathcal{C}_2$.*

Note that the above notion of immediate entailment relation slightly differs from the immediate entailment relations in [KP13]. Instead of using the semantic implication as in [KP13] we now use the provability relation with respect to the superposition calculus defined earlier. This stronger notion of immediate entailment relation was chosen in

order to assert the existence of a loop-free proof of the $\mathrm{FOL}_{\mathrm{ID}}$ sequent corresponding to the entailment[2]

Definition 3.27. *Let $\mathcal{C} = [C \mid \eta \simeq s^i x]$ be an n-clause and let $s^{r_1}x, \ldots, s^{r_k}x$ be all the maximum ω-terms (with respect to the tree-ordering) in $\mathcal{C}$ containing the variable x. Then the rank of the $\mathcal{C}$ is given by* $\mathrm{rank}(\mathcal{C}) = i - \max\{r_1, \ldots, r_k\}$.

For a natural number i and a clause set S we denote by $S[i]$ the set $\{\mathcal{C} \in S \mid \mathrm{rank}(\mathcal{C}) = i\}$.

Definition 3.28. *Let L be an n-clause language, S be an L-clause set, δ an inference relation for S, and $\sqsupseteq$ an immediate entailment relation. An inductive cycle with respect to δ and $\sqsupseteq$ is a 4-tuple $(i, j, S_{\mathsf{init}}, S_{\mathsf{loop}})$ where i, j are natural numbers with $j > 0$, $S_{\mathsf{init}} \subseteq S[i]$, $S_{\mathsf{loop}} \subseteq S[i+j]$ such that all of the conditions below are satisfied:*

- $S_{\mathsf{init}} \vdash_\delta \eta \not\approx k$ *for all* k *with* $i \leq k \leq i + j$;
- $S_{\mathsf{init}} \vdash_\delta S_{\mathsf{loop}}$;
- $S_{\mathsf{loop}} \sqsupseteq S_{\mathsf{init}}{\downarrow_j}$.

We call the first component of a cycle, the cycle's offset. Let $\vdash_\delta$ be the unrestricted inference relation and let $\sqsupseteq$ be the equality relation on clauses. Let $S = \{(\mathrm{E1}), (\mathrm{E2}), (\mathrm{E3})\}$ where (E1), (E2) and (E3) are the clauses of Example 2 and set $S_{\mathsf{init}} = \{(\mathrm{E3})\}$ and $S_{\mathsf{loop}} = \{(\mathrm{E3}){\downarrow_1}\}$. Then $(0, 1, S_{\mathsf{init}}, S_{\mathsf{loop}})$ clearly is an inductive cycle for S with respect to δ and $\sqsupseteq$.

Theorem 3.6. *Cycles are inductive loops.*

Proof. See proof of Theorem 2 in [KP13]. □

Definition 3.29 (Cyclic superposition refutation)**.** *Let S be an L-clause set, $\sqsupseteq$ an immediate entailment relation and δ an inference relation. A superposition deduction π from S is a cyclic superposition refutation w.r.t. δ and $\sqsupseteq$ if $\pi : S \vdash_\delta \eta \not\approx 0, \ldots, \pi : S \vdash_\delta \eta \not\approx i - 1$ and S admits an inductive cycle $(i, j, S_{\mathsf{init}}, S_{\mathsf{loop}})$ w.r.t. δ and $\sqsupseteq$.*

The refutational correctness of the n-clause calculus relies on the fact that for any $i \in \mathbb{N}$, the clause set $\{\eta \not\approx 0, \ldots, \eta \not\approx i - 1, \eta \prec i\}$ is unsatisfiable. Intuitively, this is equivalent to the statement that for every natural number n we have either $n = 0$, or $n = 1$, or $\ldots$, or $n = i-1$ or $n > i$. To complete our running example we now have refuted the clause set $\{(\mathrm{E1}), (\mathrm{E2}), (\mathrm{E3})\}$ in the n-clause calculus, since $(0, 1, \{(\mathrm{E3})\}, \{(\mathrm{E3})\}{\downarrow_1})$ is an inductive cycle for this set.

[2]The restriction on the immediate entailment relation chosen here is probably too strong. In [KP13] clause subsumption is an immediate entailment relation, but it is not an immediate entailment relation in the sense of Definition 3.26. It would suffice to require that the entailment between suitable translations of the clauses are $\mathcal{C}$ and $\mathcal{D}$ is provable in LK.

CHAPTER 4

First-Order Logic with Inductive Definitions

In this section we present the logic $\mathrm{FOL_{ID}}$ that extends the usual first-order logic by inductively defined predicates. This logic, together with the semantics and the calculi presented in this section, was introduced by Brotherston and Simpson in [BS10]. The logic $\mathrm{FOL_{ID}}$ serves as the target formalism for the translation from the n-clause logic. In Section 4.1 we first introduce in detail the syntax of this logic. In Section 4.2 will see three increasingly strong notions of semantics of $\mathrm{FOL_{ID}}$. Section 4.3 presents several calculi for this logic and discusses some of their properties and their relation to the different kinds of semantics. Finally in Section 4.4 we shall shortly discuss the discrepancy between the calculi LKID and CLKID^{ω} and why it is of importance for our translation of n-clause refutations to inductive proofs.

4.1 Syntax

Let us again start by defining the signature of a $\mathrm{FOL_{ID}}$ language. As for the n-clause logic a signature is a structure which contains all of the non-logical symbols used by a logic. In the case of first-order logic with inductive definitions we will need to distinguish between the term-forming symbols (i.e. variables and function constants), predicate symbols and the so-called inductively defined predicate symbols.

Definition 4.1 (Signature)**.** *A signature is a triple* $\Sigma = (\Sigma_{\mathsf{func}}, \Sigma_{\mathsf{ord}}, \Sigma_{\mathsf{ind}})$*, where* Σ_{func} *is a countable set of function symbols with their respective arities,* Σ_{ord} *is a countable set of predicate symbols with their respective arities and* Σ_{ind} *is a finite set of predicate symbols with their respective arities such that* $\Sigma_{\mathsf{ord}} \cap \Sigma_{\mathsf{ind}} = \varnothing$*.*

The predicate symbols in Σ_{ord} and Σ_{ind} are called *ordinary* predicate symbols and *inductive* predicate symbols, respectively. Inductive predicates will be treated specially

by the semantics of the logic. The semantics of inductive predicate symbols will be driven by a syntactic structure that besides the signature is part of a every $\mathrm{FOL_{ID}}$ language — the *productions*. Roughly speaking productions are rules that define operators that will be used to describe the closure of the corresponding inductive predicates.

Definition 4.2 (Inductive Definition Set)**.** *An inductive definition set Φ for a signature $\Sigma = (\Sigma_{\mathsf{func}}, \Sigma_{\mathsf{ord}}, \Sigma_{\mathsf{ind}})$ is a finite set of productions, where a production is a pair*

$$(\{Q_1(\mathbf{u_1}), \ldots, Q_h(\mathbf{u_h}), P_{j_1}(\mathbf{t_1}), \ldots, P_{j_m}(\mathbf{t_m})\}, P_i(\mathbf{t}))$$

with $Q_1, \ldots, Q_h \in \Sigma_{\mathsf{ord}}$ and $P_{j_1}, \ldots, P_{j_m}, P_i \in \Sigma_{\mathsf{ind}}$ and $\mathbf{u1}, \ldots, \mathbf{u_h}, \mathbf{t_1}, \ldots, \mathbf{t_m}, \mathbf{t}$ are sequences of terms of appropriate length.

For the sake of readability we will usually represent productions as rules of the form

$$\frac{Q_1(\mathbf{u_1}), \ldots, Q_h(\mathbf{u_h}), P_{j_1}(\mathbf{t_1}), \ldots, P_{j_m}(\mathbf{t_m})}{P_i(\mathbf{t}).}$$

The atomic formulas $Q_l(\mathbf{u_i})$ and $P_{j_k}(\mathbf{t_k})$ with $l = 1, \ldots, h$ and $k = 1, \ldots, m$ are usually referred to as the premises of the production and the formula $P_i(\mathbf{t})$ is called the production's conclusion.

Example 4. The natural numbers, even numbers, and odd numbers can be represented by inductive predicates $\mathsf{N}/1$, $\mathsf{E}/1$ and $\mathsf{O}/1$ respectively, with the mutually recursive productions below

$$\frac{}{\mathsf{N}0} \quad \frac{\mathsf{N}x}{\mathsf{N}\mathsf{s}x} \quad \frac{}{\mathsf{E}0} \quad \frac{\mathsf{O}x}{\mathsf{E}\mathsf{s}x} \quad \frac{\mathsf{E}x}{\mathsf{O}\mathsf{s}x}.$$

As we will see when defining the semantics of inductive definitions, the rules above do not define structures that are isomorphic to the natural numbers, even numbers, and odd numbers respectively, instead they just define the inductive closure with respect to the given productions. In order to obtain structures that are isomorphic to the natural numbers etc. we need to add axioms that ensure the injectivity of the interpretation of s.

Definition 4.3 (Language)**.** *A $\mathrm{FOL_{ID}}$ language L is a triple $L = (\mathcal{X}, \Sigma, \Phi)$, where Σ is a $\mathrm{FOL_{ID}}$ signature, $\mathcal{X}$ is a variable set that forms with Σ_{func} a term language, and Φ is an inductive definition set for Σ.*

For the remainder of this introduction to first-order logic with inductive definitions we shall fix a $\mathrm{FOL_{ID}}$ language $\mathcal{L} = (\mathcal{X}, \Sigma, \Phi)$ with inductive predicates $P_1/k_1, \ldots, P_n/k_n$.

Definition 4.4 (Syntax)**.** *The set $Form(L)$ of L-formulas is defined inductively as follows:*

- *if $P/n \in \Sigma_{\mathsf{ord}} \cup \Sigma_{\mathsf{ind}}$ and $t_1, \ldots, t_n \in \mathcal{T}_{\mathcal{X}}^{\Sigma_{\mathsf{func}}}$, then $P(t_1, \ldots, t_n) \in Form(L)$.*
- *if $t_1, t_2 \in \mathcal{T}_{\mathcal{X}}^{\Sigma_{\mathsf{func}}}$, then $t_1 = t_2 \in Form(L)$.*
- *if $F \in Form(L)$, then $\neg F \in Form(L)$,*

- *if* $F_1, F_2 \in Form(L)$ *and* $* \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$, *then* $F_1 * F_2 \in Form(L)$,
- *if* $x \in \mathcal{X}$ *and* $F \in Form(L)$, *then* $\forall xF, \exists xF \in Form(L)$.

Note that the $\mathrm{FOL_{ID}}$ languages defined above do not include truth constants $\top$ and $\bot$. Later it will be convenient to have these constants at our disposal. Fortunately, the logic $\mathrm{FOL_{ID}}$ has the built-in predicate constant $= /2$ which we can use to define these constants and their corresponding axioms. We define the truth constants $\top$ and $\bot$ as follows

$$\top = \forall x\ x = x$$
$$\bot = \neg\top.$$

Definition 4.5. *Let* $\mathcal{L}$ *be a* $\mathrm{FOL_{ID}}$ *language and let* F *be an* $\mathcal{L}$*-formula in prenex form. Then by* $\overline{F}$ *we denote the matrix of* F.

4.2 Semantics

We will now present three possible semantics for the logic $\mathrm{FOL_{ID}}$. First of all we present the usual first-order semantics which is used to define the more powerful notions of Henkin semantics and standard semantics.

4.2.1 First-order semantics

The first-order semantics of $\mathrm{FOL_{ID}}$ are the usual semantics of first-order logic. That is we interpret functions over first-order structures which consist of a non-empty domain and an interpretation function which assigns actual functions or predicates to the non-logical constant symbols.

Definition 4.6 (Structures and valuations)**.** *An* L*-structure* $\mathcal{M}$ *is a pair* $\mathcal{M} = (D, I)$, *where* D *is a non-empty set and* I *is a function which interprets the symbols in* Σ *as follows:*

- $f^I : D^n \to D$, *for all* $f/n \in \Sigma_{\mathsf{func}}$,
- $P^I \subseteq D^n$, *for all* $P/n \in \Sigma_{\mathsf{ord}} \cup \Sigma_{\mathsf{ind}}$.

A function $\rho : X \to D$ *is called an* $\mathcal{M}$*-valuation.*

For convenience we will sometimes denote the interpretation of a symbol c by $c^{\mathcal{M}}$ instead of c^I. If $\mathcal{M}$ is clear from the context we speak of a valuation instead of an $\mathcal{M}$-valuation. For a valuation ρ we denote by $\rho[x \mapsto m]$ with $m \in D$ the valuation that differs from ρ only for x and assigns x to m.

Definition 4.7 (Term evaluation)**.** *Let* $\mathcal{M} = (D, I)$ *an* L*-structure and* ρ *an* $\mathcal{M}$*-valuation. The term evaluating function* $[\![\cdot]\!]^{\mathcal{M}}_{\rho} : \mathcal{T}_{\mathcal{X}}^{\Sigma_{\mathsf{func}}} \to D$ *is defined as follows:*

- $[\![x]\!]^{\mathcal{M}}_{\rho} = \rho(x)$, *for all* $x \in \mathcal{X}$,
- $[\![f(t_1, \ldots, t_n)]\!]^{\mathcal{M}}_{\rho} = f^I([\![t_1]\!]^{\mathcal{M}}_{\rho}, \ldots, [\![t_n]\!]^{\mathcal{M}}_{\rho})$, *for all* $f/n \in \Sigma_{\mathsf{func}}$, *and* $t_1, \ldots, t_n \in \mathcal{T}_{\mathcal{X}}^{\Sigma_{\mathsf{func}}}$.

We extend the term evaluation to sequences of terms by applying the evaluation function pointwise. For a sequence of terms $\mathbf{t} = (t_1, \ldots, t_n)$ we define

$$[\![\mathbf{t}]\!]^{\mathcal{M}}_{\rho} = ([\![t_1]\!]^{\mathcal{M}}_{\rho}, \ldots, [\![t_n]\!]^{\mathcal{M}}_{\rho}).$$

Definition 4.8 (First-order semantics)**.** *Let* $\mathcal{M} = (D, I)$ *be an* L*-structure and* ρ *an* $\mathcal{M}$*-valuation. The satisfaction relation* $\models$ *is defined as follows*

- $\mathcal{M} \models_{\rho} P(\mathbf{t})$ *if* $[\![\mathbf{t}]\!]^{\mathcal{M}}_{\rho} \in P^I$,
- $\mathcal{M} \models_{\rho} t_1 = t_2$ *if* $[\![t_1]\!]^{\mathcal{M}}_{\rho} = [\![t_2]\!]^{\mathcal{M}}_{\rho}$
- $\mathcal{M} \models_{\rho} \neg F_1$ *if* $\mathcal{M} \not\models_{\rho} F_1$,
- $\mathcal{M} \models_{\rho} F_1 \vee F_2$ *if* $\mathcal{M} \models_{\rho} F_1$ *or* $\mathcal{M} \models_{\rho} F_2$,
- $\mathcal{M} \models_{\rho} F_1 \wedge F_2$ *if* $\mathcal{M} \models_{\rho} F_1$ *and* $\mathcal{M} \models_{\rho} F_2$,
- $\mathcal{M} \models_{\rho} F_1 \to F_2$ *if* $\mathcal{M} \not\models_{\rho} F_1$ *or* $\mathcal{M} \models_{\rho} F_2$,
- $\mathcal{M} \models_{\rho} \forall x F$ *if* $\mathcal{M} \models_{\rho[x \mapsto d]} F$, *for every* $d \in D$,
- $\mathcal{M} \models_{\rho} \exists x F$ *if* $\mathcal{M} \models_{\rho[x \mapsto d]} F$, *for some* $d \in D$.

It is easy to see that our definitions of the truth constants $\top$ and $\bot$ are adequate with respect to the first-order semantics, that is, $\top$ is valid and $\bot$ is unsatisfiable.

Under standard semantics inductive predicates behave just as ordinary predicates. In the following we will consider two notions of semantics which specialize the first-order semantics by restricting the structures to those under which the inductive predicates do behave "more" like actual inductively defined predicates.

4.2.2 Standard semantics

The standard semantics of an inductive predicate is the least fixed point of the productions that define the predicate. These semantics are obtained from first-order semantics by restricting the structures to those that interpret the inductively defined predicates accordingly.

Usually inductive sets are defined by giving a base set and iterating a monotone operator until a fixed point is reached (possibly after ω steps). Since the productions of the inductive predicates may be mutually recursive we need to define an operator which operates on each predicate simultaneously. At each stage the operator closes the approximated sets under the productions.

Definition 4.9 (Definition set operator)**.** *Let $\mathcal{M} = (D, I)$ be an L-structure. We define the partition $\Phi_1, \ldots, \Phi_n$ of Φ by*

$$\Phi_i = \{\phi \in \Phi \mid P_i \text{ is the inductive predicate in the conclusion of } \phi\}.$$

Let each set of productions Φ_i be indexed by r with $1 \leq r \leq |\Phi_i|$ and for each production $\Phi_{i,r}$ of the form

$$\Phi_{i,r} = \frac{Q_1(\mathbf{u_1}), \ldots, Q_h(\mathbf{u_h}), P_{j_1}(\mathbf{t_1}), \ldots, P_{j_m}(\mathbf{t_m})}{P_i(\mathbf{t})}$$

we define the function $\varphi_{i,r} : \mathcal{P}(D^{k_1}) \times \cdots \times \mathcal{P}(D^{k_n}) \to \mathcal{P}(D^{k_i})$ by

$$\varphi_{i,r}(X_1, \ldots, X_n) = \{[\![\mathbf{t}]\!]^{\mathcal{M}}_{\rho} \mid [\![\mathbf{u_1}]\!]^{\mathcal{M}}_{\rho} \in Q_1^{\mathcal{M}}, \ldots, [\![\mathbf{u_h}]\!]^{\mathcal{M}}_{\rho} \in Q_h^{\mathcal{M}}, \\ [\![\mathbf{t_1}]\!]^{\mathcal{M}}_{\rho} \in X_{j_1}, \ldots, [\![\mathbf{t_m}]\!]^{\mathcal{M}}_{\rho} \in X_{j_m}, \rho \text{ is a valuation}\}.$$

Then we define the function $\varphi_i : \mathcal{P}(D^{k_1}) \times \cdots \times \mathcal{P}(D^{k_n}) \to \mathcal{P}(D^{k_i})$ by

$$\varphi_i(X_1, \ldots, X_n) = \bigcup_{r=1}^{|\Phi_i|} \varphi_{i,r}(X_1, \ldots, X_n).$$

Finally, the definition set operator $\varphi_\Phi : \mathcal{P}(D^{k_1}) \times \cdots \times \mathcal{P}(D^{k_n}) \to \mathcal{P}(D^{k_1}) \times \cdots \times \mathcal{P}(D^{k_n})$ is given by

$$\varphi_\Phi(X_1, \ldots, X_n) = (\varphi_1(X_1, \ldots, X_n), \ldots, \varphi_n(X_1, \ldots, X_n)).$$

We denote by π_i^n the i-th projection function for n-tuples.

Definition 4.10 (Approximants)**.** *Let $\mathcal{M} = (D, I)$ be an L-structure. Define the indexed set $(\varphi_\Phi^\alpha \subseteq \mathcal{P}(D^{k_1}) \times \cdots \times \mathcal{P}(D^{k_n}))_{0 \leq \alpha \leq \omega}$ by $\varphi_\Phi^\alpha = \bigcup_{\beta<\alpha} \varphi_\Phi(\varphi_\Phi^\beta)$. The set $\pi_i^\alpha(\varphi_\Phi^\alpha)$ with $0 \leq \alpha \leq \omega$ is called the α-th $\mathcal{M}$-approximant of P_i and we will denote this set by $P_i^{(\alpha,\mathcal{M})}$.*

Definition 4.11 (Standard Structure)**.** *An L-structure $\mathcal{M}$ is said to be standard if $P_i^{\mathcal{M}} = \bigcup_{0 \leq \alpha \leq \omega} P_i^{(\alpha,\mathcal{M})}$ for all $i \in \{1, \ldots, n\}$.*

The standard semantics of $\mathrm{FOL_{ID}}$ formulas are then the usual semantics but restricted to the case of standard structures.

Example 5. Consider a $\mathrm{FOL_{ID}}$ language consisting of constants $\mathsf{0}/0$, $\mathsf{s}/1$ the inductive predicate symbol $\mathsf{N}/1$ and the productions $/\mathsf{N0}$ and $\mathsf{N}x/\mathsf{N}\mathsf{s}x$. Then the structure $\mathcal{M}$ with domain $\{0, 1\}$ and $\mathsf{0}^{\mathcal{M}} = 0$, $\mathsf{s}^{\mathcal{M}}(0) = 0$, $\mathsf{s}^{\mathcal{M}}(1) = 0$ is a standard structure and $\mathsf{N}^{\mathcal{M}} = \{0\}$. In order to obtain that N is bijective to $\mathbb{N}$ it suffices to consider structures that satisfy the formulas $\forall x \mathsf{0} \neq \mathsf{s}x$ and $\forall x \forall y(\mathsf{s}x = \mathsf{s}y \to x = y)$.

Brotherston and Simpson showed in [BS10] that it is possible to embed true arithmetic in $\mathrm{FOL_{ID}}$ with respect to standard semantics (see Lemma 3.11 in [BS10]). Therefore the set of sentences valid under standard semantics is not recursively enumerable.

4.2.3 Henkin Semantics

In the following we will define the notion of Henkin semantics. These semantics are not directly used in this thesis but they turn out to give a concrete semantic characterization of the formulas that are provable in the proof system LKID for structural induction. Henkin semantics of $\mathrm{FOL}_{\mathrm{ID}}$ are inspired by Henkin's approach to obtain completeness for higher-order calculi by relaxing the semantics. Here we will interpret inductively defined predicates with respect to Henkin classes. Roughly speaking Henkin semantics drop the condition "the smallest set such that ..." that usually is the first part of an inductive definition.

Definition 4.12 (Henkin Class)**.** *Let $\mathcal{M}$ be a structure with domain D. A Henkin class for $\mathcal{M}$ is a sequence of sets $\mathcal{H} = (H_k)_{k\geq 0}$ such that for each $k \in \mathbb{N}$ the following conditions are satisfied*

$$
\begin{aligned}
&\{(d,d) \mid d \in D\} \in H_2\\
&\text{if } Q/k \text{ is a predicate symbol, then } Q^{\mathcal{M}} \in H_k\\
&\text{if } R \in H_{k+1}, \text{ and } d \in D, \text{ then } \{(d_1,\ldots,d_k) \mid (d_1,\ldots,d_n,d)\} \in H_k\\
&\text{if } R \in H_k \text{ and } t_1(x_1,\ldots,x_m),\ldots,t_k(x_1,\ldots,x_m) \text{ are terms, then}\\
&\qquad\qquad \{(d_1,\ldots,d_m) \mid ([\![t_1]\!]^{\mathcal{M}}_{\mathbf{x}\mapsto\mathbf{d}},\ldots,[\![t_k]\!]^{\mathcal{M}}_{\mathbf{x}\mapsto\mathbf{d}})\} \in H_m\\
&\text{if } R \in H_k, \text{ then } D^k \setminus R \in H_k\\
&\text{if } R_1, R_2 \in H_k, \text{ then } R_1 \cap R_2 \in H_k\\
&\text{if } R \in H_{k+1}, \text{ then } \{(d_1,\ldots,d_k) \mid \exists d \text{ s.t. } (d_1,\ldots,d_k,d) \in R\} \in H_k.
\end{aligned}
$$

Definition 4.13. *Let $\mathcal{M}$ be a structure and let $\mathcal{H} = (H_k)_{k\geq 0}$ be a Henkin class. A tuple $(X_1,\ldots,X_n) \in \mathcal{P}(D^{k_1}) \times \ldots \mathcal{P}(D^{k_n})$ is an $\mathcal{H}$-point if $X_i \in H_{k_i}$ for all $i \in \{1,\ldots,n\}$.*

A fixed $\mathcal{H}$-point is an $\mathcal{H}$-point that is this a fixed point of the definition set operator φ_Φ.

Definition 4.14. *A Henkin structure is a pair $(\mathcal{M},\mathcal{H})$ where $\mathcal{M}$ is a structure and $\mathcal{H}$ is a Henkin class for $\mathcal{M}$ such that there exists a list fixed $\mathcal{H}$-point $\mu_{\mathcal{H}} \cdot \varphi_\Phi$ of φ_Φ, and $P_i^{\mathcal{M}} = \pi_i^n(\mu_{\mathcal{H}} \cdot \varphi_\Phi)$ for each $i \in \{1,\ldots,n\}$.*

It is not hard to see that see that every standard structure $\mathcal{M}$ with domain D together with $\mathcal{H} = (\mathcal{P}(D^k))_{k\geq 0}$ is a Henkin structure. The following lemma will help us to understand how Henkin semantics relate to the induction rules.

Lemma 4.1. *Let $\mathcal{H} = (H_k)_{k\geq 0}$ for a structure $\mathcal{M}$, ρ be a valuation, and F a formula and $x_1,\ldots,x_k$ pairwise distinct variables. Then we have*

$$\{(d_1,\ldots,d_k) \mid \mathcal{M} \models_{\rho[x_1\mapsto d_1,\ldots,x_k\mapsto d_k]} F\} \in H_k.$$

Proof. See proof of Lemma 2.7 in [BS10]. □

The following example is intended to give the reader an impression about the relation between Henkin semantics and structural induction. We anticipate here the notion of induction axioms; for a more general definition of these axioms see Definition 4.21. By a structural induction axiom over our usual natural number predicate N we understand a formula of the form

$$F(0) \wedge \forall x(F(x) \to F(sx)) \to \forall x(\mathsf{N}x \to F(x)),$$

where F is any formula. For a given formula $F(x)$ we denote the corresponding induction axiom by $\mathrm{I}_x F$.

Example 6. For simplicity let us consider a language consisting only of the usual natural number predicate $\mathsf{N}/1$ with its function symbols and productions. Let $F(x)$ be a formula, $(\mathcal{M}, \mathcal{H})$ a Henkin structure with domain D, and let ρ be any valuation. Assume $(\mathcal{M}, \mathcal{H}) \models_\rho F(\mathsf{0})$ and $(\mathcal{M}, \mathcal{H}) \models_\rho \forall x(F(x) \to F(sx))$. We define $[\![F]\!]^{\mathcal{M}}_\rho = \{d \in D \mid \mathcal{M} \models_{\rho[x \mapsto d]} F\}$. It is then not hard to see that the set $[\![F]\!]^{\mathcal{M}}_\rho$ is a fixed point of φ_{N}, where φ_{N} is the definition set operator for the predicate symbol N. By Lemma 4.1 we moreover have $[\![F]\!]^{\mathcal{M}}_\rho \in H_1$. In other words the set $[\![F]\!]^{\mathcal{M}}_\rho \in H_1$ is a fixed $\mathcal{H}$-point. Since $\mathsf{N}^{\mathcal{M}}$ is the least fixed $\mathcal{H}$-point and H_1 is closed under intersection it is the case that $[\![F]\!]^{\mathcal{M}}_\rho \supseteq \mathsf{N}^{\mathcal{M}}$. But this means that $(\mathcal{M}, \mathcal{H}) \models_{\rho[x \mapsto d]} \mathsf{N}x \to F$ for all $d \in D$. Therefore $(\mathcal{M}, \mathcal{H}) \models \mathrm{I}_x F$ i.e. the structural induction axioms are valid with respect to Henkin semantics.

4.3 Sequent Calculi for $\mathrm{FOL_{ID}}$

In the following we will describe the calculi $\mathsf{LK}, \mathsf{LKID}, \mathsf{LKID}^\omega$ and CLKID^ω for $\mathrm{FOL_{ID}}$ as introduced in [BS10]. We start with LK which is a basic calculus for first-order logic with equality. Secondly we will describe the calculus LKID which is a calculus for the formalization of arguments by structural induction. Finally we will introduce the systems LKID^ω and CLKID^ω – the former being an infinitary proof system and the latter being a restriction thereof. Both calculi formalize arguments by infinite descent.

4.3.1 LK

The proof system LK is very close to the usual Gentzen style sequent calculus for classical first-order logic, except that our version of LK includes equality. It serves as the base system from which the stronger systems $\mathsf{LKID}, \mathsf{LKID}^\omega$, and CLKID^ω are developed. Note also that our sequents are pairs of sets of formulas, instead of pairs of sequences of formulas. Hence there is no need for exchange rules and contraction rules and some inferences may be expressed in a slightly different way.

Definition 4.15. *A sequent is a pair of the form $\Gamma \Rightarrow \Delta$ where Γ and Δ are finite sets.*

In the context of sequents juxtaposition of sets of formulas by "," indicates union of sets.

Definition 4.16. *The calculus* LK *consists of the following inference rules.*
Structural rules and axioms:

$$\frac{}{\Gamma \Rightarrow \Delta, \top}\ (\mathrm{Ax}\top) \qquad \frac{}{\Gamma \Rightarrow \Delta}\ \Gamma \cap \Delta \neq \varnothing\ (\mathrm{Ax})$$

$$\frac{}{\bot, \Gamma \Rightarrow \Delta}\ (\mathrm{Ax}\bot) \qquad \frac{\Gamma \Rightarrow F, \Delta \qquad \Pi, F \Rightarrow \Lambda}{\Gamma, \Pi \Rightarrow \Delta, \Lambda}\ (\mathrm{Cut})$$

$$\frac{\Gamma' \Rightarrow \Delta'}{\Gamma \Rightarrow \Delta}\ \Gamma' \subseteq \Gamma, \Delta' \subseteq \Delta\ (\mathrm{Wk}) \qquad \frac{\Gamma \Rightarrow \Delta}{\Gamma[\theta] \Rightarrow \Delta[\theta]}\ (\mathrm{Subst})$$

Logical rules:

$$\frac{\Gamma \Rightarrow F, \Delta}{\Gamma, \neg F \Rightarrow \Delta}\ (\neg\mathrm{L}) \qquad \frac{\Gamma, F \Rightarrow \Delta}{\Gamma \Rightarrow \neg F, \Delta}\ (\neg\mathrm{R})$$

$$\frac{\Gamma, F \Rightarrow \Delta \qquad \Pi, G \Rightarrow \Lambda}{\Gamma, \Pi, F \vee G \Rightarrow \Delta, \Lambda}\ (\vee\mathrm{L}) \qquad \frac{\Gamma \Rightarrow F, G, \Delta}{\Gamma \Rightarrow F \vee G, \Delta}\ (\vee\mathrm{R})$$

$$\frac{\Gamma, F, G \Rightarrow \Delta}{\Gamma, F \wedge G \Rightarrow \Delta}\ (\wedge\mathrm{L}) \qquad \frac{\Gamma \Rightarrow F, \Delta \qquad \Pi \Rightarrow G, \Lambda}{\Gamma, \Pi \Rightarrow F \wedge G, \Delta, \Lambda}\ (\wedge\mathrm{R})$$

$$\frac{\Gamma \Rightarrow F, \Delta \qquad \Pi, G \Rightarrow \Lambda}{\Gamma, \Pi, F \to G \Rightarrow \Delta, \Lambda}\ (\to\mathrm{L}) \qquad \frac{\Gamma, F \Rightarrow G, \Delta}{\Gamma \Rightarrow F \to G, \Delta}\ (\to\mathrm{R})$$

$$\frac{\Gamma, F[x/t] \Rightarrow \Delta}{\Gamma, \forall x F \Rightarrow \Delta}\ (\forall\mathrm{L}) \qquad \frac{\Gamma \Rightarrow F, \Delta}{\Gamma \Rightarrow \forall x F, \Delta}\ x \notin \mathrm{FV}(\Gamma \cup \Delta)\ (\forall\mathrm{R})$$

$$\frac{\Gamma, F \Rightarrow \Delta}{\Gamma, \exists x F \Rightarrow \Delta}\ x \notin \mathrm{FV}(\Gamma \cup \Delta)\ (\exists\mathrm{L}) \qquad \frac{\Gamma \Rightarrow F[x/t], \Delta}{\Gamma \Rightarrow \exists x F, \Delta}\ (\exists\mathrm{R})$$

$$\frac{\Gamma[x/u, y/t] \Rightarrow \Delta[x/u, y/t]}{\Gamma[x/t, y/u], t = u \Rightarrow \Delta[x/t, y/u]}\ (=\mathrm{L}) \qquad \frac{}{\Gamma \Rightarrow t = t, \Delta}\ (=\mathrm{R})$$

A derivation of LK is a rooted tree whose nodes are annotated by sequents such that every inner node v satisfies the following: The sequent associated with v is the conclusion of a rule instance whose premise(s) is (are) the sequent(s) associated to v's child (children). A proof of LK is a derivation of LK all of whose nodes are the conclusion of an inference rule of LK.

The substitution rule is usually not part of the calculus LK. This rule does not add any power to the calculus as it can be simulated by the other rules, but it turns out to be a useful abbreviation when dealing with cyclic proofs. The axioms for the truth constants $\top$ and $\bot$ are also not part of the systems defined in [BS10] but by the definitions of these constants it is easy to see that these rules are redundant and can be eliminated.

The calculus LK is sound and complete with respect to first-order semantics.

4.3.2 LKID

The calculus LKID is a calculus for the formalization of arguments by structural induction. It extends the inference rules of LK by right-introduction rules for inductive predicates and left-introduction rules for inductive predicates. The left-introduction rules represent the actual induction rules.

Definition 4.17 (Right-introduction rules for inductive predicates)**.** *For each production* $\phi \in \Phi$ *of the form*

$$\phi = \frac{Q_1(\mathbf{u_1}), \ldots, Q_h(\mathbf{u_h}), P_{j_1}(\mathbf{t_1}), \ldots, P_{j_m}(\mathbf{t_m})}{P_i(\mathbf{t})}$$

the right-introduction rule $P_i\mathrm{R}$ *is of the form*

$$\frac{\Gamma \Rightarrow Q_1\mathbf{u_1}(\mathbf{u}), \Delta \ \ldots \ \Gamma \Rightarrow Q_h\mathbf{u_h}(\mathbf{u}), \Delta \ \ldots \ \Gamma \Rightarrow P_{j_1}\mathbf{t_1}(\mathbf{u}), \Delta \ \ldots \ \Gamma \Rightarrow P_{j_m}\mathbf{t_m}(\mathbf{u}), \Delta}{\Gamma \Rightarrow P_i\mathbf{t}(\mathbf{u}), \Delta,} \ (P_i\mathrm{R})$$

where for every $\mathbf{r} \in \{\mathbf{u_1}, \ldots, \mathbf{u_h}, \mathbf{t_1}, \ldots, \mathbf{t_m}, \mathbf{u}\}$ *the expression* $\mathbf{r}(\mathbf{u})$ *represents* $\mathbf{r}[\mathbf{x}/\mathbf{u}]$ *with* $\mathbf{x}$ *being the vector of variables explicitly identified as occurring in the production* φ*.*

Before we will begin with the definition of the left-introduction rules let us look at the shape of the induction schemes in order to get an intuition about the more complicated mutually recursive predicates. For the usual definition of the natural numbers this is straightforward: there will be a base case and a step case as described in Section 1.1. The induction scheme thus looks as follows.

$$\frac{\vdash \varphi(\mathsf{0}) \qquad \varphi(\alpha) \vdash \varphi(\mathsf{s}\alpha)}{\vdash \forall x \varphi(x)}$$

The situation is a little bit more complicated in the presence of mutual dependencies between predicates. Consider again the definition of the even numbers and odd numbers given in Example 4. In order to prove that a property φ_{E} holds for all even numbers we start as usual with the number 0, but the induction step is more involved since we have to "jump over" the odd numbers. The idea is to introduce a suitable property φ_{O} for the odd numbers such that we can show that if φ_{E} holds for an even number x then φ_{O} holds for the next odd number, and vice-versa. Then it remains to show the base case, that is, φ_{E} holds for $\mathsf{0}$. We thus have an induction scheme that looks as follows

$$\frac{\vdash \varphi_{\mathsf{E}}\mathsf{0} \qquad \varphi_{\mathsf{E}}\alpha \vdash \varphi_{\mathsf{O}}\mathsf{s}\alpha \qquad \varphi_{\mathsf{O}}\alpha \vdash \varphi_{\mathsf{E}}\mathsf{s}\alpha}{\vdash \forall_E x \varphi_{\mathsf{E}}(x)}$$

We thus need additional induction hypotheses for each mutually dependent inductive predicate. In the following definition we formalize this notion of predicates being mutually dependent.

Definition 4.18 (Mutual dependency)**.** *Define the relation Prem to be the least relation over the inductive predicate symbols such that $Prem(P_i, P_j)$ holds if and only if P_i occurs in the conclusion of a production in Φ and P_j occurs among the premises of the production. Define $Prem^*$ to be the reflexive and transitive closure of Prem. Then we say that two predicate symbols P_i and P_j are mutually dependent if both $Prem^*(P_i, P_j)$ and $Prem^*(P_j, P_i)$ are true.*

Definition 4.19 (Induction rules)**.** *Let P_j be an inductive predicate. We associate with every inductive predicate P_i of arity k_i a k_i-tuple $\mathbf{z_i}$ of pairwise distinct variables; we will refer to these variables as* induction variables*. Furthermore we associate with each inductive predicate P_i an arbitrary formula F_i, possibly containing variables of $\mathbf{z}_i$ which we will call the* induction hypothesis*. For every $i \in \{1, \ldots, n\}$ define the formula G_i by*

$$G_i = \begin{cases} F_i & \text{if } P_i \text{ and } P_j \text{ are mutually dependent} \\ P_i(\mathbf{z_i}) & \text{otherwise.} \end{cases}$$

An instance of the left introduction rule for the inductive predicate P_j follows the schema below:

$$\frac{\textit{minor premises} \qquad \Gamma, F_j\mathbf{u} \Rightarrow \Delta}{\Gamma, P_j\mathbf{u} \Rightarrow \Delta}\,(\mathrm{Ind}P_j)$$

The sequent $\Gamma, F_j\mathbf{u} \Rightarrow \Delta$ is called the major premise*. For each production of Φ containing in its conclusion a predicate P_i that is mutually dependent with P_j, say:*

$$\frac{Q_1\mathbf{u_1}(\mathbf{x}), \ldots, Q_h\mathbf{u_h}(\mathbf{x}), P_{j_1}\mathbf{t_1}(\mathbf{x}), \ldots, P_{j_m}\mathbf{t_m}(\mathbf{x})}{P_i\mathbf{t}(\mathbf{x})}$$

there is a corresponding minor premise

$$\Gamma, Q_1\mathbf{u_1}(\mathbf{y}), \ldots, Q_h\mathbf{u_h}(\mathbf{y}), G_{j_1}\mathbf{t_1}(\mathbf{y}), \ldots, G_{j_m}\mathbf{t_m}(\mathbf{y}) \Rightarrow F_i\mathbf{t}(\mathbf{y}).$$

The vector $\mathbf{y}$ is a vector of distinct variables of the length of $\mathbf{x}$ such that $y \notin \mathrm{FV}(\Gamma \cup \Delta \cup \{P_j\mathbf{u}\})$ for all $y \in \mathbf{y}$. In other words $\mathbf{y}$ is a vector of eigenvariables.

Taking again the rules introduced in Example 4, and let $F_{\mathsf{N}}(x)$ be an arbitrary formula, then the induction rule for natural numbers looks as follows:

$$\frac{\Gamma \Rightarrow F_{\mathsf{N}}\mathsf{0}, \Delta \qquad \Gamma, F_{\mathsf{N}}\nu \Rightarrow F_{\mathsf{N}}\mathsf{s}\nu, \Delta \qquad \Gamma, F_{\mathsf{N}}t \Rightarrow \Delta}{\Gamma, \mathsf{N}t \Rightarrow \Delta.}\,(\mathrm{Ind}\mathsf{N})$$

For the even numbers the induction rule looks as shown below:

$$\frac{\Gamma \Rightarrow F_{\mathsf{E}}\mathsf{0}, \Delta \qquad \Gamma, F_{\mathsf{E}}\nu \Rightarrow F_{\mathsf{O}}\mathsf{s}\nu, \Delta \qquad \Gamma, F_{\mathsf{O}}\nu \Rightarrow F_{\mathsf{E}}\mathsf{s}\nu, \Delta \qquad \Gamma, F_{\mathsf{E}}t \Rightarrow \Delta}{\Gamma, \mathsf{E}t \Rightarrow \Delta,}\,(\mathrm{Ind}\mathsf{E})$$

where $F_{\mathsf{E}}(x), F_{\mathsf{O}}(x)$ are arbitrary formulas. The induction rules of LKID are slightly unusual in the sense that they include a major premise. The major premise has two roles: first of all it relates the induction hypothesis to the domain represented by the inductive predicate, this shall become clearer when we consider the corresponding induction axioms; and secondly the major premise also plays the role of non-free cut (see [Bus98]) that introduces a possibly non-analytic induction hypothesis. Therefore the formulation of induction rules with major premises also leads to a more elegant formulation of the free-cut elimination theorem (see [Bus98]), which for this reason is simply called the cut-elimination theorem for LKID.

For languages containing a predicate that is similar to the predicate N in the sense that there are only the two productions /N0 and Nx/Nsx for that predicate, we assume that there are induction rules that allow for a larger induction step. In other words we assume that for every $j > 0$ there are rules of the form

$$\frac{\Gamma \Rightarrow F_{\mathsf{N}}(0), \Delta \quad \ldots \quad \Gamma \Rightarrow F_{\mathsf{N}}(\overline{j-1}), \Delta \quad \Gamma, F_{\mathsf{N}}x \Rightarrow F_{\mathsf{N}}\mathsf{s}^j x, \Delta \quad \Gamma, F_{\mathsf{N}}t \Rightarrow \Delta}{\Gamma, \mathsf{N}t \Rightarrow \Delta}\,(\mathrm{Ind}^j\mathsf{N})$$

It is not hard to see that adding these rules to LKID does not increase the system's strength. Indeed we can replace an (Ind^{j}N) inference by an (IndN) inference that uses the induction hypothesis $\wedge_{i=0}^{j-1} F_{\mathsf{N}}\mathsf{s}^i x$. Let $\pi^0_{\mathsf{base}}, \ldots, \pi^{j-1}_{\mathsf{base}}$, π_{step} and π_{cut} be the proofs of the minor and major premises of the (Ind^{j}N) inference. Then we prove the base case as follows:

$$\frac{\begin{array}{ccc}\vdots\pi^0_{\mathsf{base}} & & \vdots\pi^{j-1}_{\mathsf{base}} \\ \Gamma, F_{\mathsf{N}}0 \Rightarrow \Delta & \ldots & \Gamma, F_{\mathsf{N}}\overline{j-1} \Rightarrow \Delta\end{array}}{\Gamma \Rightarrow \bigwedge_{i=0}^{j-1} F_{\mathsf{N}}\bar{i}, \Delta.}\,(\wedge\mathrm{R}^*)$$

The step case is proved as shown below:

$$\frac{\begin{array}{ccccc} & & & & \vdots\pi_{\mathsf{step}} \\ F_{\mathsf{N}}\mathsf{s}^1 x \Rightarrow F_{\mathsf{N}}\mathsf{s}^1 x & \ldots & F_{\mathsf{N}}\mathsf{s}^{j-1}x \Rightarrow F_{\mathsf{N}}\mathsf{s}^{j-1}x & & \Gamma, F_{\mathsf{N}}x \Rightarrow F_{\mathsf{N}}\mathsf{s}^j x, \Delta\end{array}}{\Gamma, \bigwedge_{i=0}^{j-1} F_{\mathsf{N}}s^i x \Rightarrow \bigwedge_{i=0}^{j-1} F_{\mathsf{N}}s^{i+1}x, \Delta.}\,(\wedge\mathrm{R}^*)$$

Finally the major premise can be proved as follows:

$$\frac{\begin{array}{c}\vdots\pi_{\mathsf{cut}} \\ \Gamma, F_{\mathsf{N}}t \Rightarrow \Delta\end{array}}{\Gamma, \bigwedge_{i=0}^{j-1} F_{\mathsf{N}}\mathsf{s}^i t \Rightarrow \Delta}\,(\wedge\mathrm{L}^*, \mathrm{Wk})$$

We will now introduce the induction axioms corresponding to the induction rules that we formulated above. These induction axioms will later serve us to represent the inductive information that is required to prove a sequent in a canonical way. We can then much easier express the provability of a sequent with respect to some inductive theory.

Definition 4.20. *We denote by* LKID-I*, the proof system obtained from* LKID *by removing the induction rules. We denote by* $\mathsf{LKID}(\Sigma_1)$*, the proof system obtained from* LKID *by restricting the induction rules to* Σ_1*-induction invariants.*

Definition 4.21 (Induction Axioms)**.** *Let* P_j *be an inductive predicate,* $P_{l_1}, \ldots, P_{l_o}$ *be the predicates that are mutually dependent with* P_j *and let* $F_{l_1}, \ldots, F_{l_o}$ *be their respective induction hypotheses. Let furthermore* $\phi_1, \ldots, \phi_r$ *be the productions containing in their conclusion a predicate symbol that is mutually dependent with* P_j*. We use the same notations as in Definition 4.19. Let* $k \in \{1, \ldots, r\}$*, then the production* ϕ_k *is of the form*

$$\frac{Q_1\mathbf{u_1}(\mathbf{x}), \ldots, Q_h\mathbf{u_h}(\mathbf{x}), P_{j_1}\mathbf{t_1}(\mathbf{x}), \ldots, P_{j_m}\mathbf{t_m}(\mathbf{x})}{P_i\mathbf{t}(\mathbf{x})}.$$

The formula ICase_k *is given by*

$$\forall \mathbf{x}\left(\bigwedge_{i=1}^{h} Q_i\mathbf{u_i}(\mathbf{x}) \wedge \bigwedge_{i=1}^{m} G_{j_i}\mathbf{t_i}(\mathbf{x}) \rightarrow F_i\mathbf{t}(\mathbf{x})\right)$$

The induction axiom $\mathrm{I}^{P_j}_{\mathbf{z_j}}(F_{l_1}, \ldots, F_{l_n})$ *is given by*

$$\bigwedge_{i=1}^{r} \mathsf{ICase}_r \rightarrow \forall \mathbf{z_j}(P_j\mathbf{z_j} \rightarrow F_j\mathbf{z_j}).$$

Example 7. Let $F_{\mathsf{N}}(x)$ be an arbitrary formula, then the induction axiom $\mathrm{I}^{\mathsf{N}}_x F_{\mathsf{N}}$ is given by

$$F_{\mathsf{N}}0 \wedge \forall x(F_{\mathsf{N}}x \rightarrow F_{\mathsf{N}}\mathsf{s}x) \rightarrow \forall x(\mathsf{N}x \rightarrow F_{\mathsf{N}}x).$$

Note that the major premise of the induction rules manifests itself in the induction axioms as the relativization of the quantifier to the domain N.

We will denote by $\mathrm{Ind}_{\mathsf{N}}$ the set of induction axioms for the predicate N. In the following we will sketch a proof showing that the theory $\mathrm{Ind}_{\mathsf{N}}$ is "correct" with respect to the induction rule for the predicate N. This means we will show that every formula in $\mathrm{Ind}_{\mathsf{N}}$ is provable in LKID using only induction rules for the predicate N, and furthermore we will show that the formulas in Ind simulate the induction rules for the predicate N. In a general setting the correctness of the axioms as formulated above can be shown in an analogous way.

Let $\mathrm{I}^{\mathsf{N}}_x F_{\mathsf{N}}$ be any induction axiom in $\mathrm{Ind}_{\mathsf{N}}$, then we can prove it in LKID as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{F_{\mathsf{N}}z \Rightarrow F_{\mathsf{N}}z \qquad F_{\mathsf{N}}\mathsf{s}z \Rightarrow F_{\mathsf{N}}\mathsf{s}z}{F_{\mathsf{N}}z \rightarrow F_{\mathsf{N}}\mathsf{s}z, F_{\mathsf{N}}z \Rightarrow F_{\mathsf{N}}\mathsf{s}z}(\rightarrow\mathrm{L})}{\forall x(F_{\mathsf{N}}x \rightarrow F_{\mathsf{N}}\mathsf{s}x), F_{\mathsf{N}}z \Rightarrow F_{\mathsf{N}}\mathsf{s}z}(\forall\mathrm{L}) \qquad F_{\mathsf{N}}x \Rightarrow F_{\mathsf{N}}x}{F_{\mathsf{N}}0, \forall x(F_{\mathsf{N}}x \rightarrow F_{\mathsf{N}}\mathsf{s}x), \mathsf{N}x \Rightarrow F_{\mathsf{N}}x}(\mathrm{Ind}\mathsf{N})}{F_{\mathsf{N}}0, \forall x(F_{\mathsf{N}}x \rightarrow F_{\mathsf{N}}\mathsf{s}x) \Rightarrow \mathsf{N}x \rightarrow F_{\mathsf{N}}x}(\rightarrow\mathrm{R})}{F_{\mathsf{N}}0, \forall x(F_{\mathsf{N}}x \rightarrow F_{\mathsf{N}}\mathsf{s}x) \Rightarrow \forall x(\mathsf{N}x \rightarrow F_{\mathsf{N}}x)}(\forall\mathrm{R})}{\Rightarrow F_{\mathsf{N}}0 \wedge \forall x(F_{\mathsf{N}}x \rightarrow F_{\mathsf{N}}\mathsf{s}x) \rightarrow \forall x(\mathsf{N}x \rightarrow F_{\mathsf{N}}x).}(\rightarrow\mathrm{R}, \wedge\mathrm{L})$$

Now it remains to show that every instance of an induction rule for the predicate N in an arbitrary proof π can be simulated by an induction axiom. We proceed by induction on the number of induction inferences in the proof π. The base case is trivial. Consider now an uppermost induction inference I. The inference I must necessarily be of the form:

$$I\ \frac{\begin{array}{ccc}\vdots\pi_{\mathsf{base}} & \vdots\pi_{\mathsf{step}} & \vdots\pi_{\mathsf{cut}}\\ \Gamma \Rightarrow F\mathsf{0}, \Delta & \Gamma, Fx \Rightarrow F\mathsf{s}x, \Delta & \Gamma, Ft \Rightarrow \Delta\end{array}}{\Gamma, \mathsf{N}t \Rightarrow \Delta.}\ (\mathrm{Ind}\mathsf{N})$$

By the induction hypothesis we obtain proofs $\pi'_{\mathsf{base}}, \pi'_{\mathsf{step}}$, and π'_{cut} of the sequents $\Lambda_1, \Gamma \Rightarrow F\mathsf{0}, \Delta$; $\Lambda_2, \Gamma, Fx \Rightarrow F\mathsf{s}x, \Delta$ and $\Lambda_3, \Gamma, Ft \Rightarrow \Delta$, respectively, where Λ_1, Λ_2, and Λ_3 are finite subsets of $\mathrm{Ind}_{\mathsf{N}}$. We can simulate the induction inference I by using the induction axiom $\mathrm{I}^{\mathsf{N}}_x F$ as shown below

$$\frac{\dfrac{\begin{array}{cc}\vdots\pi'_{\mathsf{base}} & \dfrac{\vdots\pi'_{\mathsf{step}}}{\Lambda_2, \Gamma, Fx \Rightarrow F\mathsf{s}x, \Delta}\\ \Lambda_1, \Gamma, \Rightarrow F\mathsf{0}, \Delta & \Lambda_2, \Gamma, \Rightarrow \forall x(Fx \to F\mathsf{s}x), \Delta\end{array}}{\Lambda_1, \Lambda_2, \Gamma, \Rightarrow F\mathsf{0} \wedge \forall x(Fx \to F\mathsf{s}x), \Delta} \quad \begin{array}{c}\vdots\pi'_{\mathsf{cut}}\\ \Lambda_3, \Gamma, Ft \Rightarrow \Delta\end{array}}{\Lambda_1, \Lambda_2, \Lambda_3, \mathrm{I}_x F, \Gamma \Rightarrow \Delta}\ (\to\mathrm{L})$$

In the article [BS10] Brotherston and Simpson prove several properties of the calculus LKID. In particular the calculus LKID is shown to be sound and complete with respect to Henkin semantics and moreover it is shown that LKID admits cut-elimination.

4.3.3 LKID^ω

In this section we will present the infinitary proof system LKID^ω – a calculus for reasoning by infinite descent in $\mathrm{FOL_{ID}}$ – as originally introduced by Brotherston and Simpson in [BS10]. Even though we are not using this very calculus in the analysis of Peltier's cyclic superposition calculus, introducing the calculus LKID^ω is required for the later definition of CLKID^ω.

The rules of the proof system LKID^ω are the rules of LKID with the exception that the left introduction rules $\mathrm{Ind}P_i$, where P_i is an inductive predicate, are replaced by the *case-split* rule. The case-split rules represent simple case distinctions according to the productions of an inductive predicate.

Definition 4.22 (Case-split rules)**.** *Let P_i be an inductive predicate, then the corresponding case-split rule* $\mathrm{Case}P_i$ *is of the form:*

$$\frac{\textit{case distinctions}}{\Gamma, P_i\mathbf{t}(\mathbf{u}) \Rightarrow \Delta}\ (\mathrm{Case}P_i)$$

where for every production $\phi \in \Phi$ having P_i in its conclusion, say:

$$\frac{Q_1\mathbf{u_1}, \ldots, Q_h\mathbf{u_h}, P_{j_1}\mathbf{t_1}, \ldots, P_{j_m}\mathbf{t_m}}{P_i\mathbf{t}}$$

there is a corresponding case distinction

$$\Gamma, \mathbf{u} = \mathbf{t}(\mathbf{y}), Q_1\mathbf{u_1}(\mathbf{y}), \ldots, Q_h\mathbf{u_h}(\mathbf{y}), P_{j_1}\mathbf{t_1}(\mathbf{y}), \ldots, P_{j_m}\mathbf{t_m}(\mathbf{y}) \Rightarrow \Delta$$

where $\mathbf{y}$ is a vector of eigenvariables i.e. the variables in $\mathbf{y}$ are pairwise distinct and $y \notin \mathrm{FV}(\Gamma \cup \Delta \cup \{P_i\mathbf{t}(\mathbf{u})\})$ for all $y \in \mathbf{y}$.

For the even numbers and the odd numbers with the usual productions of Example 4 the corresponding case split rules are as follows:

$$\frac{\Gamma, t = \mathsf{s}x, \mathsf{E}x \Rightarrow \Delta}{\Gamma, \mathsf{O}t \Rightarrow \Delta}\ (\mathrm{CaseO})$$

$$\frac{\Gamma, t = 0 \Rightarrow \Delta \qquad \Gamma, t = \mathsf{s}x, \mathsf{O}x \Rightarrow \Delta}{\Gamma, \mathsf{E}t \Rightarrow \Delta}\ (\mathrm{CaseE})$$

The proof system LKID^ω is based on infinite derivation trees. Derivations in the proof system LKID^ω correspond to the infinite trees described by the inferences rules. In such an infinite derivation tree we distinguish between *leaf-nodes* and *bud-nodes* – a leaf is the conclusion of an inference rule which has no premises; a bud is any non-leaf node which is not a conclusion of an inference rule. We can now introduce the notion of LKID^ω pre-proofs.

Definition 4.23 (Pre-Proof)**.** *An LKID^ω pre-proof of a sequent $\Gamma \Rightarrow \Delta$ is a possibly infinite derivation tree π formed according to the inference rules of LKID^ω such that the root of π is $\Gamma \Rightarrow \Delta$, and π does not contain buds.*

It is not hard to see that LKID_ω pre-proofs are not sound. This is because infinite paths are intended to represent arguments by infinite descent, though we have not introduced any notion that ensures the progression of the arguments. Hence some infinite branches represent arguments which do not progress i.e. the measure associated to this argument does not strictly decrease. In order to constrain the set of pre-proofs to the set of sound pre-proofs we need to introduce the concepts of traces and progression points.

Given an LKID^ω pre-proof π we define a path to be a finite or infinite sequence of sequents $(S_i)_{0 \leq \beta < \alpha}$ where $\alpha \in \mathbb{N} \cup \{\omega\}$ such that S_{i+1} is a child of S_i in π for all i with $i + 1 < \alpha$.

Definition 4.24 (Trace)**.** *Let π be an LKID^ω pre-proof and let $(\Gamma_i \Rightarrow \Delta_i)_{i \geq 0}$ be a path in π. A trace following $(\Gamma_i \Rightarrow \Delta_i)_{i \geq 0}$ is a sequence of formulas $(\tau_i)_{i \geq 0}$ such that, for all i, the following hold:*

- *$\tau_i = P_{j_i}\mathbf{t_i}$, where $j_i \in \{1, \ldots, n\}$;*

- *if* $\Gamma_i \Rightarrow \Delta_i$ *is the conclusion of a* Subst *rule then* $\tau_i = \tau_{i+1}[\theta]$*, where* θ *is the substitution associated with the rule instance;*
- *if* $\Gamma_i \Rightarrow \Delta_i$ *is the conclusion of a* $=\mathrm{L}$ *rule with principal formula* $t = u$*, then there is a formula* F *and variables* x, y *such that* $\tau_i = F[x/t, y/u]$ *and* $\tau_{i+1} = F[x/u, y/t]$*;*
- *if* $\Gamma_i \Rightarrow \Delta_i$ *is the conclusion of a case-split rule then either* $\tau_{i+1} = \tau_i$ *or* τ_{i+1} *is the principal formula of the rule instance and* τ_{i+1} *is a case-descendant of* τ_i*. In the latter case,* i *is said to be a progress point of the trace;*
- *if* $\Gamma_i \Rightarrow \Delta_i$ *is the conclusion of any other rule, then* $\tau_{i+1} = \tau_i$*.*

An infinitely progressing trace is a trace having infinitely many progress points.

We will always underline the traced formulas so that the a proof is more easily seen to satisfy the trace condition.

Definition 4.25. *An* LKID^ω *pre-proof* π *is an* LKID^ω *proof if it satisfies the* global trace condition *— that is, every infinite path* $(\Gamma_i \Rightarrow \Delta_i)_{i\geq 0}$ *in* π *admits an infinitely progressing trace following the path* $(\Gamma_i \Rightarrow \Delta_i)_{i\geq k}$ *for some* $k \geq 0$*.*

Brotherston and Simpson have shown in [BS10] that the calculus LKID^ω is sound and cut-free complete with respect to standard semantics. This calculus is thus very powerful but it is not useful for formal proofs because its proof objects are in general infinite.

4.3.4 CLKID^ω

In this section we will introduce another proof system, namely CLKID^ω, as originally presented by Brotherston and Simpson [BS10]. The proof system CLKID^ω is a subsystem of LKID^ω which arises by restricting the proofs of LKID^ω to those having a regular tree shape. Proofs of CLKID^ω can be represented as finite graphs and more precisely as sequent calculus derivation whose buds are connected to inner sequents – the so-called companions. Intuitively, a cyclic proof represents the LKID^ω proof obtained by constantly unfolding the buds by the proof rooted in their companions.

Definition 4.26 (Companion)**.** *Let* B *be a bud in an* LKID^ω *derivation tree* π*. An internal node* C *of* π *is a companion of* B *if* B *and* C *are labelled with the same sequent.*

Definition 4.27 (CLKID^ω pre-proof)**.** *A* CLKID^ω *pre-proof* π *of a sequent* $\Gamma \Rightarrow \Delta$ *is a pair* $(\gamma, \mathcal{R})$ *where* γ *is a finite* LKID^ω *derivation tree whose root is* $\Gamma \Rightarrow \Delta$*, and* $\mathcal{R}$ *is a function assigning to each bud node of* γ *a companion in* γ*.*

The graph of π*, denoted* $\mathcal{G}_\pi$ *is obtained by identifying each bud* B *in* γ *with its companion node* $\mathcal{R}(B)$*.*

Definition 4.28 (CLKID^ω proof)**.** *A* CLKID^ω *proof is a* CLKID^ω *pre-proof whose graph satisfies the global trace condition.*

Example 8. Consider a language consisting of function symbols $0/0$, $\mathsf{s}/1$, inductive predicate symbols $\mathsf{N}/1, \mathsf{E}/1$, $\mathsf{O}/1$, and the productions given in Example 4. The derivation below is a CLKID^ω proof of the sequent $\mathsf{E}x \vee \mathsf{O}x \Rightarrow \mathsf{N}x$. The symbols $(\circlearrowleft_i)$ with $i \in \{1, 2\}$ connect buds with their companion.

$$\dfrac{\dfrac{\dfrac{}{\Rightarrow \mathsf{N}0}\,(\mathsf{NR})}{x = 0 \Rightarrow \mathsf{N}x}\,(=\!\mathrm{L}) \quad \dfrac{\dfrac{\dfrac{\underline{\mathsf{O}x} \Rightarrow \mathsf{N}x\ (\circlearrowleft_1)}{\underline{\mathsf{O}y} \Rightarrow \mathsf{N}y}\,(\mathsf{Subst})}{\underline{\mathsf{O}y} \Rightarrow \mathsf{N}\mathsf{s}y}\,(\mathsf{NR}_2)}{x = \mathsf{s}y, \underline{\mathsf{O}y} \Rightarrow \mathsf{N}x}\,(=\!\mathrm{L})}{\underline{\mathsf{E}x} \Rightarrow \mathsf{N}x\ (\circlearrowleft_2)}\,(\mathsf{CaseE}) \quad \dfrac{\dfrac{\dfrac{\underline{\mathsf{E}x} \Rightarrow \mathsf{N}x\ (\circlearrowleft_2)}{\underline{\mathsf{E}y} \Rightarrow \mathsf{N}y}\,(\mathsf{Subst})}{\underline{\mathsf{E}y} \Rightarrow \mathsf{N}\mathsf{s}y}\,(\mathsf{NR}_2)}{x = \mathsf{s}y, \underline{\mathsf{E}y} \Rightarrow \mathsf{N}x}\,(=\!\mathrm{L})}{\underline{\mathsf{O}x} \Rightarrow \mathsf{N}x\ (\circlearrowleft_1)}\,(\mathsf{CaseO})}{\mathsf{E}x \vee \mathsf{O}x \Rightarrow \mathsf{N}x}\,(\vee\mathrm{L})$$

Brotherston and Simpson have shown in [BS10] that the system CLKID^ω is complete with respect to Henkin semantics. The question whether CLKID^ω is also sound with respect to Henkin semantics was left open until Berardi and Tatsuta provided a negative result in [BT17a]. We will see more about this in Section 4.4.

4.4 The Brotherston-Simpson Conjecture

It is a natural question to ask how the provability in LKID relates to that of the system CLKID^ω. Brotherston and Simpson showed that every sequent provable in LKID is also provable in CLKID^ω by representing induction inferences by cycles.

Theorem 4.2. *If a sequent S is provable in* LKID*, then S is provable in* CLKID^ω*.*

Proof. See proof of Theorem 7.6 in [BS10]. □

Furthermore, Brotherston and Simpson conjectured that the reverse is also true. We refer to this conjecture as the Brotherston-Simpson conjecture. Since the calculus LKID is sound and complete with respect to Henkin semantics this conjecture is equivalent to the conjecture that the proof system CLKID^ω is sound with respect to Henkin semantics. In 2017 Berardi and Tatsuta proved that this is indeed not the case by providing a Henkin countermodel to a sequent that is provable in CLKID^ω [BT17a]. The soundness of the system LKID with respect to Henkin semantics implies the unprovability of the sequent. Berardi and Tatsuta showed that a variant of the Hydra statement is not provable in LKID.

The Hydra statement was introduced by Kirby and Paris in [KP82]. This formal statement about the termination of a particular reduction system is inspired by the mythological creature of the same name that grows two new heads for each of its heads that is chopped off. Formally, a hydra is a tree whose leaves represent the hydra's heads. Whenever a leaf is removed from the hydra, then the remaining subtree rooted in the leaf's parent is replicated n times at the next lower level where n is the number of cut off heads. The Hydra-statement then asserts that every possible strategy to cut off heads will eventually terminate, i.e. eventually only the root node remains.

The 2-Hydra statement considered by Berardi and Tatsuta in [BT17a] restricts the hydra statement to the existence of terminating reduction strategy for 2-hydras, i.e. hydras having always exactly two heads. The underlying reduction system can then be expressed by the length of the path to the heads as follows:

$$\begin{aligned}&\text{if } n \geq 1 \text{ and } m \geq 2 \text{ then } (n, m) \mapsto (n-1, m-2)\\&\text{if } n \geq 2 \text{ then } (n, 0) \mapsto (n-1, n-2)\\&\text{if } m \geq 2 \text{ then } (0, m) \mapsto (m-1, m-2).\end{aligned}$$

The hydra is considered dead if no more transformations are applicable, that is, if the hydra is of one of the forms: (0,0), (1, 0), or (n, 1) with $n \geq 1$.

The 2-Hydra statement can be formalized in first-order logic with inductive definitions using a language consisting of function symbols $\mathsf{0}/0$, $\mathsf{s}/1$; an ordinary predicate symbol $\mathsf{p}/2$ and an inductive predicate symbol $\mathsf{N}/1$ with the productions $/\mathsf{N0}$ and $\mathsf{N}x/\mathsf{N}\mathsf{s}x$. Let the formulas H_a, H_b, H_c, H_d be given by

$$\begin{aligned}H_a &= \forall x(\mathsf{N}x \to \mathsf{p}(\mathsf{0},\mathsf{0}) \wedge \mathsf{p}(\overline{1},\mathsf{0}) \wedge \mathsf{p}(x,\overline{1})),\\H_b &= \forall x \forall y(\mathsf{N}x \wedge \mathsf{N}y \to (\mathsf{p}(x,y) \to \mathsf{p}(\mathsf{s}x,\mathsf{ss}y))),\\H_c &= \forall y(\mathsf{N}y \to (\mathsf{p}(\mathsf{s}y,y) \to \mathsf{p}(\mathsf{0},\mathsf{ss}y))),\\H_d &= \forall x(\mathsf{N}x \to (\mathsf{p}(\mathsf{s}x,x) \to \mathsf{p}(\mathsf{ss}x,\mathsf{0}))).\end{aligned}$$

The formal 2-Hydra statement H is the sequent given by

$$H = H_a, H_b, H_c, H_d \Rightarrow \forall x \forall y(\mathsf{N}x \wedge \mathsf{N}y \to \mathsf{p}(x,y)).$$

As already mentioned above, Berardi and Tatsuta showed that the sequent H is provable in CLKID^ω ant that it is not provable in LKID because there exists a Henkin countermodel. Interestingly, the sequent H becomes provable in system LKID with respect to the axiom $\forall x(\mathsf{N}x \to \mathsf{0} \neq \mathsf{s}x)$ if the language is enhanced by an additional inductive predicate symbol $\leq$ with the following productions

$$\frac{}{x \leq x} \quad \text{and} \quad \frac{x \leq y}{x \leq \mathsf{s}y}.$$

This means the proof system LKID is not conservative with respect to the addition of inductive predicate symbols. This observation is important for us since we will later translate languages of the n-clause logic to languages of $\mathrm{FOL}_{\mathrm{ID}}$ that extend the language by inductive predicate symbols. Moreover we try to relate the refutability in the n-clause calculus to the provability in LKID. Thus, it is interesting to observe that the choice of the inductive predicates may possibly affect this relation. Indeed we shall later consider in Section 7.2.2 an n-clause set whose translation apparently exhibits similar properties to the sequent H given above.

It was recently shown by Simpson in [Sim17] that cyclic arithmetic is equivalent to Peano arithmetic. This result was then generalized by Berardi and Tatsuta who showed in [BT17b] that the discrepancy between the proof systems LKID and CLKID^ω breaks down, if a suitable fragment of arithmetic is added to the systems.

CHAPTER 5

Translation of n-Clause Logic to $\mathrm{FOL}_{\mathrm{ID}}$

This section describes the translation of the n-clause logic to first-order logic with inductive definitions. In Section 5.1 we will describe the target $\mathrm{FOL}_{\mathrm{ID}}$ language that is chosen to carry out the translation from the n-clause logic. Section 5.2 describes the translation of n-clauses to $\mathrm{FOL}_{\mathrm{ID}}$ formulas. Finally in Section 5.3 we will show that the translation satisfies several desirable semantic properties.

5.1 The Language

The choice of the target $\mathrm{FOL}_{\mathrm{ID}}$ language is mainly motivated by the characteristics of the n-clause logic as well as its semantics described in Section 3.1.2. Since the n-clause logic is two-sorted we need two predicates to differentiate between the domains of these two sorts. Moreover the sort ω ranges over the natural numbers so the corresponding predicate needs to be inductively defined. Since the sort ι ranges over the set of ι-terms, the predicate representing this sort is inductively defined as well. These inductive predicates are driven by the productions induced by the term constructors and their respective types.

Definition 5.1. *Let $L = (\Sigma, \mathcal{X})$ be an n-clause language. We define the* $\mathrm{FOL}_{\mathrm{ID}}$ *language $\mathcal{L}_L$ as $\mathcal{L}_L = (\mathcal{X}_{\mathcal{L}_L}, \Sigma_{\mathcal{L}_L}, \Phi)$, where $\mathcal{X}_{\mathcal{L}_L} = \mathcal{X} \cup \{\eta\}$, and $\Sigma_{\mathcal{L}_L} = (\Sigma, \varnothing, \{\mathsf{T}_\iota/1, \mathsf{T}_\omega/1\})$ and the set of productions Φ is given by:*

$$\Phi = \left\{ \frac{\mathsf{T}_{s_1} x_1 \ \ldots \ \mathsf{T}_{s_n} x_n}{\mathsf{T}_{s_{n+1}} f(x_1, \ldots, x_n)} \,\middle|\, f : \langle s_1, \ldots, s_n, s_{n+1} \rangle \in \Sigma \right\}.$$

From now on we consider that the languages L and $\mathcal{L}_L$ are fixed. Since we are mainly interested in carrying out inductions on the natural numbers, it is arguable whether it

is necessary to introduce productions for the ι terms. It may be sufficient to introduce closure axioms for the predicate T_ι with respect to the ι-term constructors. However using these productions for ι-terms mimics the semantics of the n-clause logic quite closely; this eases the later semantic analysis of the translation.

Let us consider the language of the clauses of Example 2. The language contains the symbols $\mathsf{p} : \omega \to \iota \to \iota, \mathsf{g} : \iota \to \iota, \mathsf{t} : \iota$. Hence the corresponding first-order language will contain the function symbols $\mathsf{0}/0, \mathsf{s}/1, \mathsf{p}/2, \mathsf{g}/1, \mathsf{t}/1$ and the following productions:

$$\frac{}{\mathsf{T}_\omega \mathsf{0}} \quad \frac{\mathsf{T}_\omega x}{\mathsf{T}_\omega \mathsf{s} x} \quad \frac{\mathsf{T}_\omega x \ \mathsf{T}_\iota y}{\mathsf{T}_\iota \mathsf{p}(x,y)} \quad \frac{\mathsf{T}_\iota x}{\mathsf{T}_\iota \mathsf{g} x} \quad \frac{}{\mathsf{T}_\iota \mathsf{t}}.$$

The constants $\mathsf{0} : \omega$ and $\mathsf{s} : \omega \to \omega$ are the only constants of range ω in the n-clause language L. Thus the set of productions Φ contains only the two productions $/\mathsf{T}_\omega \mathsf{0}$ and $\mathsf{T}_\omega x / \mathsf{T}_\omega \mathsf{s} x$ having the predicate symbol T_ω in their conclusion. Hence the predicate T_ω behaves analogously to the predicate N defined in Section 4. We need two axioms to ensure that the predicate T_ω with the functions $\mathsf{0}$ and s and its productions actually defines a structure similar to the natural numbers.

Definition 5.2. *The set of formulas* $\mathrm{T}^{\mathsf{inj}}$ *is given by*

$$\begin{aligned} \mathsf{inj}^{\mathsf{0}} &= \forall x \ \mathsf{s}(x) \neq \mathsf{0}, \\ \mathsf{inj}^{\mathsf{s}} &= \forall x \forall y (\mathsf{s}(x) = \mathsf{s}(y) \to x = y), \\ \mathrm{T}^{\mathsf{inj}} &= \{ \ \mathsf{inj}^{\mathsf{0}}, \ \mathsf{inj}^{\mathsf{s}} \ \}. \end{aligned}$$

5.2 Constraint Clauses

The translation of n-clauses is inspired by the implicative semantics defined in Section 3.1.2. We represent n-clauses as implications whose antecedent consists of the conjunction of the atoms in the clause's constraint and whose succedent is the disjunction of the literals in the clause part. Moreover we will need to relativize the variables to their respective domains. Finally, we close the formula universally.

Definition 5.3. *Let* $\mathcal{C} = [C \mid X]$ *be an* L*-constraint clause. Then we define the following* $\mathcal{L}_L$ *formulas:*

$$\begin{aligned} \lceil \mathcal{C} \rceil_{\mathsf{type}} &= \bigwedge_{x : s \in var(\mathcal{C})} \mathsf{T}_s x, \\ \lceil \mathcal{C} \rceil_{\mathsf{constraint}} &= \bigwedge_{\eta \simeq t \in X} \eta \simeq t, \\ \lceil \mathcal{C} \rceil_{\mathsf{clause}} &= \bigvee_{t_1 \dot{\simeq} t_2} t_1 \doteq t_2, \\ \lceil \mathcal{C} \rceil &= \forall_{x \in var(\mathcal{C})} \left(\lceil \mathcal{C} \rceil_{\mathsf{type}} \wedge \lceil \mathcal{C} \rceil_{\mathsf{constraint}} \to \lceil \mathcal{C} \rceil_{\mathsf{clause}} \right) \end{aligned}$$

We will sometimes implicitly use an equivalent formulation of the translation defined above which replaces implications by negation and disjunction. It is easy to see that this translation preserves logical equivalence with respect to first-order semantics. The translation defined above naturally extends to sets of n-clauses by taking the conjunction of the translated clauses. The translations of the clauses (E1), (E2) and (E3) of Example 2 are respectively given by

$$\top \to \mathsf{p}(\mathsf{0},\mathsf{t}) = \mathsf{t},$$
$$\forall x \forall y(\mathsf{T}_\omega x \wedge \mathsf{T}_\iota y \to \mathsf{p}(x,y) \neq \mathsf{t} \vee \mathsf{p}(\mathsf{s}x, \mathsf{g}y) = \mathsf{t}),$$
$$\forall x \forall y(\mathsf{T}_\omega x \wedge \mathsf{T}_\iota y \wedge \eta = x \to \mathsf{p}(x,y) \neq \mathsf{t}).$$

5.3 Semantic Analysis

Our objective is to translate n-clause refutations into proofs of CLKID^ω and LKID. For prospective applications we might be interested in translating proofs of LKID with a certain type of induction invariants back to n-clause refutations. Therefore we need to make sure that the translation above is adequate with respect to these goals. That is, we need the translation to be satisfiability equivalent and moreover we would like the translation to preserve validity.

5.3.1 Satisfiability Equivalence

In the following we will show that a clause set and its translation are indeed satisfiability equivalent. We prove the satisfiability equivalence by showing that models of one formalism translate into models of the other. For one direction we show that for a given n-clause interpretation we obtain a structure for the corresponding $\mathrm{FOL}_{\mathrm{ID}}$ language by trivially extending the congruence relation to non-well typed terms. The factors of this extended congruence relation then induce a standard $\mathcal{L}_L$ structure. For the other direction we observe that equality induces an equivalence relation on ground terms.

Definition 5.4. *Let $\mathcal{I}$ be an L-interpretation. Then the relation $\stackrel{\scriptscriptstyle\vee}{\equiv}$ is given by:*

$$\stackrel{\scriptscriptstyle\vee}{\equiv} \; = \equiv^{\mathcal{I}} \cup \{(t,t) : t \in \mathit{Gnd}(\mathcal{T}^\Sigma_{\mathcal{X}})\}.$$

Furthermore we define the $\mathcal{L}_L$-structure $\mathcal{M}^{\mathcal{I}} = (D^{\mathcal{I}}, I^{\mathcal{I}})$ by:

$$D^{\mathcal{I}} = \mathit{Gnd}(\mathcal{T}^\Sigma_{\mathcal{X}})/\stackrel{\scriptscriptstyle\vee}{\equiv},$$
$$\mathsf{T}_\iota^{\mathcal{M}^{\mathcal{I}}} = \mathit{Gnd}(\mathbb{T}^\Sigma_{\mathcal{X}}(\iota))/\stackrel{\scriptscriptstyle\vee}{\equiv},\ \mathsf{T}_\omega^{\mathcal{M}^{\mathcal{I}}} = \mathit{Gnd}(\mathbb{T}^\Sigma_{\mathcal{X}}(\omega))/\stackrel{\scriptscriptstyle\vee}{\equiv},\ \textit{and}$$
$$f^{\mathcal{M}^{\mathcal{I}}}([t_1]_{\stackrel{\scriptscriptstyle\vee}{\equiv}}, \ldots, [t_n]_{\stackrel{\scriptscriptstyle\vee}{\equiv}}) = [f(t_1, \ldots, t_n)]_{\stackrel{\scriptscriptstyle\vee}{\equiv}},$$

for all $[t_1]_{\stackrel{\scriptscriptstyle\vee}{\equiv}}, \ldots, [t_n]_{\stackrel{\scriptscriptstyle\vee}{\equiv}} \in D^{\mathcal{I}}$ and every $f/n \in \Sigma$.

In order to satisfy our needs the structures defined above must satisfy two properties. First of all they need to be standard structures and secondly they need to preserve satisfiability.

Lemma 5.1. *Let $\mathcal{I}$ be an L-interpretation, then $\mathcal{M}^{\mathcal{I}}$ is a well-defined standard $\mathcal{L}_L$-structure.*

Proof. First of all for the well-definedness of $\mathcal{M}^{\mathcal{I}}$ observe that $\stackrel{\scriptscriptstyle\wedge}{\equiv}$ is a congruence relation on $Gnd(\mathcal{T}^{\Sigma}_{\mathcal{X}})$. Hence the factor set $D^{\mathcal{I}}$ is well-defined. Moreover we have

$$\stackrel{\scriptscriptstyle\wedge}{\equiv}\!\!\big|_{\left(Gnd(\mathbb{T}^{\Sigma}_{\mathcal{X}}(\iota))\cup Gnd(\mathbb{T}^{\Sigma}_{\mathcal{X}}(\omega))\right)^2} = \equiv^{\mathcal{I}},$$

which implies that the factor sets $\mathsf{T}^{\mathcal{M}^{\mathcal{I}}}_{\iota}$ and $\mathsf{T}^{\mathcal{M}^{\mathcal{I}}}_{\omega}$ are also well-defined. Since $\stackrel{\scriptscriptstyle\wedge}{\equiv}$ is a congruence relation the interpretation of the function symbols are well-defined. $\mathcal{M}^{\mathcal{I}}$ is thus clearly an $\mathcal{L}_L$ structure. It remains to show that $\mathcal{M}^{\mathcal{I}}$ is a standard structure i.e. we have to show that the following equations hold.

$$\mathsf{T}^{\mathcal{M}^{\mathcal{I}}}_{\iota} = \bigcup_{\alpha\geq 0} \mathsf{T}^{(\alpha,\mathcal{M}^{\mathcal{I}})}_{\iota},\ \mathsf{T}^{\mathcal{M}^{\mathcal{I}}}_{\omega} = \bigcup_{\alpha\geq 0} \mathsf{T}^{(\alpha,\mathcal{M}^{\mathcal{I}})}_{\omega}.$$

We will show these two equations simultaneously. For the $\subseteq$ direction we need to show that an arbitrary $[t]_{\stackrel{\scriptscriptstyle\wedge}{\equiv}} \in \mathsf{T}^{\mathcal{M}^{\mathcal{I}}}_{s}$ with $t \in Gnd(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s))$ satisfies $[t]_{\stackrel{\scriptscriptstyle\wedge}{\equiv}} \in \bigcup_{\alpha\geq 0} \mathsf{T}^{(\alpha,\mathcal{M}^{\mathcal{I}})}_{s}$. We proceed by induction on the structure of the ground term t. For the base case assume that $t = c$ where $c : s \in \Sigma$. By definition of the language $\mathcal{L}_L$ there exists a production of the form $/\ \mathsf{T}_s c \in \Phi$. Hence already the approximant $\mathsf{T}^{(1,\mathcal{M}^{\mathcal{I}})}_{s}$ contains $[c]_{\stackrel{\scriptscriptstyle\wedge}{\equiv}}$ and therefore $[t]_{\stackrel{\scriptscriptstyle\wedge}{\equiv}} \in \bigcup_{\alpha\geq 0} \mathsf{T}^{(\alpha,\mathcal{M}^{\mathcal{I}})}_{s}$. For the induction step suppose that t is of the form $f(u_1,\ldots,u_n)$ with $u_i \in Gnd(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s_i))$ and $f : (s_1,\ldots,s_n,s) \in \Sigma$. By the induction hypothesis we have $[u_i]_{\stackrel{\scriptscriptstyle\wedge}{\equiv}} \in \bigcup_{\alpha\geq 0} \mathsf{T}^{(\alpha,\mathcal{M}^{\mathcal{I}})}_{s_i}$ i.e. $[u_i]_{\stackrel{\scriptscriptstyle\wedge}{\equiv}} \in \mathsf{T}^{(\alpha_i,\mathcal{M}^{\mathcal{I}})}_{s_i}$ for some α_i for all $i = 1,\ldots,n$. Moreover by the definition of $\mathcal{L}_L$ there exists in Φ a production of the form:

$$\frac{\mathsf{T}_{s_1}x_1\ \ldots\ \mathsf{T}_{s_n}x_n}{\mathsf{T}_{s_{n+1}}f(x_1,\ldots,x_n)}.$$

Consider the α-th approximant with $\alpha = \max_{i=1,\ldots,n}\alpha_i + 1$ and the valuation $\rho = \{x_i \mapsto [u_i]_{\stackrel{\scriptscriptstyle\wedge}{\equiv}} : i = 1,\ldots,n\}$. It is then immediate that $[t]_{\stackrel{\scriptscriptstyle\wedge}{\equiv}} \in \mathsf{T}^{(\alpha,\mathcal{M}^{\mathcal{I}})}_{s_{n+1}}$. For the $\supseteq$ direction we proceed by induction on the ordinal α. For the base case suppose that $\alpha = 0$, then $\mathsf{T}^{(0,\mathcal{M}^{\mathcal{I}})}_{s} = \varnothing$ i.e. the claim trivially holds. Now suppose that the claim holds for all ordinals less than or equal to α, and suppose furthermore that there exists $d \in \mathsf{T}^{(\alpha+1,\mathcal{M}^{\mathcal{I}})}_{s} \setminus \mathsf{T}^{(\alpha,\mathcal{M}^{\mathcal{I}})}_{s}$. Thus there exists a production of the form:

$$\frac{\mathsf{T}_{s_1}x_1\ \ldots\ \mathsf{T}_{s_n}x_n}{\mathsf{T}_{s_{n+1}}f(x_1,\ldots,x_n)},$$

with $f : (s_1,\ldots,s_n,s) \in \Sigma$, and a valuation ρ such that $[\![f(x_1,\ldots,x_n)]\!]^{\mathcal{M}^{\mathcal{I}}}_{\rho} = d$ and $\rho(x_i) \in \mathsf{T}^{(\alpha,\mathcal{M}^{\mathcal{I}})}_{s_i}$ for all $i = 1,\ldots,n$. By the induction hypothesis we have $\rho(x_i) \in Gnd(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s_i))/\stackrel{\scriptscriptstyle\wedge}{\equiv}$ for all $i = 1,\ldots,n$. Hence there exist terms $t_i \in Gnd(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s_i))$ such that $[t_i]_{\stackrel{\scriptscriptstyle\wedge}{\equiv}} = \rho(x_i)$ for all $i \in \{1,\ldots,n\}$. Then by definition of $\mathcal{M}^{\mathcal{I}}$ we have $d = [f(t_1,\ldots,t_n)]_{\stackrel{\scriptscriptstyle\wedge}{\equiv}}$ and $d \in \mathsf{T}^{\mathcal{M}^{\mathcal{I}}}_{s}$ since $f(t_1,\ldots,t_n) \in Gnd(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s))$. □

It remains to show that the structures $\mathcal{M}^{\mathcal{I}}$ preserve satisfiability.

Lemma 5.2. *Let S be an L-clause set, and $\mathcal{I}$ a model of S, then $\mathcal{M}^{\mathcal{I}}$ with any valuation ρ_η such that $\rho_\eta(\eta) = \{\overline{\eta^{\mathcal{I}}}\}$ is a model of $\mathrm{T}^{\mathsf{inj}} \wedge \mathsf{T}_\omega(\eta) \wedge \lceil S \rceil$.*

Proof. Proceed indirectly and suppose that $\mathcal{M}^{\mathcal{I}} \not\models \mathrm{T}^{\mathsf{inj}} \wedge \mathsf{T}_\omega(\eta) \wedge \lceil S \rceil$. Then there are three different cases:

1. Suppose that $\mathcal{M}^{\mathcal{I}} \not\models_{\rho_\eta} \mathsf{T}_\omega(\eta)$ i.e. $\rho_\eta(\eta) = \{\overline{\eta^{\mathcal{I}}}\} \notin \mathsf{T}_\omega^{\mathcal{M}^I}$. This is impossible since $\overline{\eta^{\mathcal{I}}} \in Gnd(\mathbb{T}^\Sigma_{\mathcal{X}}(\omega))$ and $[\overline{\eta^{\mathcal{I}}}]_{\breve{\equiv}} = \{\overline{\eta^{\mathcal{I}}}\}$.

2. Suppose that $\mathcal{M}^{\mathcal{I}} \not\models \mathrm{T}^{\mathsf{inj}}$. Then there are two cases which need to be considered.

   a) Assume that there exists a term $t \in Gnd(\mathcal{T}^\Sigma_{\mathcal{X}})$ such that for $\rho = \rho_\eta[x \mapsto [t]_{\breve{\equiv}}]$ such that $\mathcal{M}^{\mathcal{I}} \not\models_\rho \mathsf{s}(x) \neq \mathsf{0}$ i.e. $[\![\mathsf{s}(x)]\!]_\rho^{\mathcal{M}^{\mathcal{I}}} = \{\mathsf{s}(t)\} = \{\mathsf{0}\} = \mathsf{0}^{\mathcal{M}^{\mathcal{I}}}$. This is impossible since $\mathsf{s}(t) \neq \mathsf{0}$.

   b) Suppose that there exists $t_1, t_2 \in Gnd(\mathcal{T}^\Sigma_{\mathcal{X}})$ such that $\mathcal{M}^{\mathcal{I}} \not\models_\rho \mathsf{s}(x) \neq \mathsf{s}(y) \vee x = y$, where $\rho = \rho_\eta[x \mapsto [t_1]_{\breve{\equiv}}, y \mapsto [t_2]_{\breve{\equiv}}]$. This implies that $\mathsf{s}(t_1)^{\mathcal{M}^{\mathcal{I}}} = \mathsf{s}(t_2)^{\mathcal{M}^{\mathcal{I}}}$ and $t_1^{\mathcal{M}^{\mathcal{I}}} \neq t_2^{\mathcal{M}^{\mathcal{I}}}$. Hence we have $t_1 \neq t_2$ and therefore $\mathsf{s}(t_1) \neq \mathsf{s}(t_2)$. Suppose without loss of generality that t_1 is not an ω-term. Then $\mathsf{s}(t_1)$ is not well-typed and hence $\mathsf{s}(t_1)^{\mathcal{M}^{\mathcal{I}}} = \{\mathsf{s}(t_1)\}$ which implies that $\mathsf{s}(t_2) \notin \mathsf{s}(t_1)^{\mathcal{M}^{\mathcal{I}}}$ i.e. $\mathsf{s}(t_1)^{\mathcal{M}} \neq \mathsf{s}(t_2)^{\mathcal{M}}$. Contradiction! Suppose now that both t_1 and t_2 are ω-terms, then since $\equiv$ is the equality relation on ω-terms we have that $s(t_1)^{\mathcal{M}^{\mathcal{I}}} = \{\mathsf{s}(t_1)\} \neq \{\mathsf{s}(t_2)\} = \mathsf{s}(t_2)^{\mathcal{M}^{\mathcal{I}}}$.

3. Suppose that $\mathcal{M}^{\mathcal{I}} \not\models_{\rho_\eta} \lceil S \rceil$. Then by construction of $\lceil S \rceil$ there exists a constraint clause $\mathcal{C} \in S$ and a valuation ρ with $\rho(\eta) = \rho_\eta(\eta)$ such that the corresponding conjunct $\lceil \mathcal{C} \rceil$ in $\lceil S \rceil$ is not satisfied by $\mathcal{M}^{\mathcal{I}}$ and ρ. In this case we have in particular

   $$\mathcal{M}^{\mathcal{I}} \not\models_\rho \neg \mathsf{T}_s(x), \text{ i.e. } \rho(x) \in \mathsf{T}_s^{\mathcal{M}^{\mathcal{I}}} \text{ for all } x_s \in var(\mathcal{C}).$$

   By definition of $\mathcal{M}^{\mathcal{I}}$ there exists for every $x_s \in var(\mathcal{C})$ a term $t^x \in Gnd(\mathbb{T}^\Sigma_{\mathcal{X}}(s))$ such that $\rho(x) = [t^x]_{\breve{\equiv}}$.

   Let $\sigma = \{x \mapsto t^x : x \in var(\mathcal{C})\}$ and observe that σ is a well-typed ground substitution with $dom(\sigma) = var(\mathcal{C})$. Consider now any three literals $\eta \simeq t \in ctr(\mathcal{C}), u_1 \simeq u_2 \in cls(\mathcal{C})$, and $v_1 \not\simeq v_2 \in cls(\mathcal{C})$. By the assumption that $\mathcal{M}^{\mathcal{I}} \not\models_\rho \lceil \mathcal{C} \rceil$ we obtain in particular that

   $$\mathcal{M}^{\mathcal{I}} \not\models_\rho \eta \neq t, \mathcal{M}^{\mathcal{I}} \not\models_\rho u_1 = u_2, \text{ and } \mathcal{M}^{\mathcal{I}} \not\models_\rho v_1 \neq v_2.$$

   This implies that the following holds:

   $$\rho(\eta) = \rho_\eta(\eta) = \{\overline{\eta^{\mathcal{I}}}\} = [\![t]\!]_\rho^{\mathcal{M}^{\mathcal{I}}},$$
   $$[\![u_1]\!]_\rho^{\mathcal{M}^{\mathcal{I}}} \neq [\![u_2]\!]_\rho^{\mathcal{M}^{\mathcal{I}}}, \text{ and } [\![v_1]\!]_\rho^{\mathcal{M}^{\mathcal{I}}} = [\![v_2]\!]_\rho^{\mathcal{M}^{\mathcal{I}}}.$$

By Lemma 5.4 we obtain that

$$\{\overline{\eta^{\mathcal{I}}}\} = [\![t]\!]_{\rho}^{\mathcal{M}^{\mathcal{I}}} = t\sigma^{\mathcal{M}^{\mathcal{I}}} \text{ i.e. } \overline{\eta^{\mathcal{I}}} = t\sigma,$$
$$u_1\sigma^{\mathcal{M}^{\mathcal{I}}} \neq u_2\sigma^{\mathcal{M}^{\mathcal{I}}}, \text{ and } v_1\sigma^{\mathcal{M}^{\mathcal{I}}} = v_2\sigma^{\mathcal{M}^{\mathcal{I}}}.$$

Then by the definition of $\mathcal{M}^{\mathcal{I}}$ we have

$$u_1\sigma \not\equiv^{\mathcal{I}} u_2\sigma, \text{ and } v_1\sigma \equiv^{\mathcal{I}} v_2\sigma,$$

which means that $\mathcal{I} \not\models u_1\sigma \simeq u_2\sigma$ and $\mathcal{I} \not\models v_1\sigma \not\simeq v_2\sigma$. Hence $\mathcal{I} \not\models \mathcal{C}\sigma$ but this contradicts the assumption that $\mathcal{I}$ is a model of S.

□

Now it remains to show the other direction. That is if $\mathsf{T}_\omega\eta \wedge \mathrm{T}^{\mathsf{inj}} \wedge \lceil S \rceil$ is satisfiable, then S is satisfiable.

Proposition 5.3. *Let S be an clause set. If the formula $\mathsf{T}_\omega\eta \wedge \mathrm{T}^{\mathsf{inj}} \wedge \lceil S \rceil$ is satisfiable with respect to standard structures, then S is also satisfiable.*

We prove the above proposition by showing that the L-interpretation induced by the terms that evaluate to the same element is a satisfying interpretation of the clause set if the original $\mathcal{L}_L$ structure is a standard model. In order to prove this proposition we first need to show several properties of terms with respect to standard $\mathcal{L}_L$-structures.

Lemma 5.4. *Let $\mathcal{M}$ be an $\mathcal{L}_L$ structure, $t \in \mathcal{T}_{\mathcal{X}}^{\Sigma}$. Then for every ground substitution σ with $var(t) \subseteq dom(\sigma)$ and every valuation ρ with $\rho|_{dom(\sigma)} = (\cdot)^{\mathcal{M}} \circ \sigma$ we have $t\sigma^{\mathcal{M}} = [\![t]\!]_{\rho}^{\mathcal{M}}$.*

Proof. The proof proceeds by induction on the structure of the term t. The base cases correspond to the cases where t is a variable or a constant. Suppose thus that $t = c$ where $c/0 \in \Sigma$. Then for all ground substitutions σ and every valuation ρ we have

$$c\sigma^{\mathcal{M}} = c^{\mathcal{M}} = [\![c]\!]_{\rho}^{\mathcal{M}}.$$

Suppose now that $t = x$ where $x \in \mathcal{X}$ and let σ be a ground substitution with $x \in dom(\sigma)$ and let ρ be any valuation with $\rho(x) = x\sigma^{\mathcal{M}}$. Then we clearly have:

$$x\sigma^{\mathcal{M}} = \rho(x) = [\![x]\!]_{\rho}^{\mathcal{M}}.$$

For the induction step suppose that t is of the form $t = f(t_1, \ldots, t_n)$ and let σ be any ground substitution. Moreover let ρ be any valuation with $\rho(x) = x\sigma^{\mathcal{M}}$ for all $x \in dom(\sigma)$. Then we have $var(t_i) \subseteq dom(\sigma)$ and hence by the induction hypothesis we have $t_i\sigma^{\mathcal{M}} = [\![t_i]\!]_{\rho}^{\mathcal{M}}$ for all $i = 1, \ldots, n$. Hence the following equations holds:

$$\begin{aligned} f(t_1, \ldots, t_n)\sigma^{\mathcal{M}} &= f(t_1\sigma, \ldots, t_n\sigma)^{\mathcal{M}} = f^{\mathcal{M}}(t_1\sigma^{\mathcal{M}}, \ldots, t_n\sigma^{\mathcal{M}}) \\ &= f^{\mathcal{M}}([\![t_1]\!]_{\rho}^{\mathcal{M}}, \ldots, [\![t_n]\!]_{\rho}^{\mathcal{M}}) = [\![f(t_1, \ldots, t_n)]\!]_{\rho}^{\mathcal{M}}. \end{aligned}$$

□

Lemma 5.5. *Let $\mathcal{M}$ be a standard $\mathcal{L}_L$ structure. Then for all $s \in \{\iota, \omega\}$ and $t \in Gnd(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s))$ we have $t^{\mathcal{M}} \in \mathsf{T}^{\mathcal{M}}_s$.*

Proof. The proof proceeds by induction on the structure of the term t. For the base case suppose that $t = c$ where $c : s \in \Sigma$. Since $\mathcal{M}$ is a standard $\mathcal{L}_L$ structure we have:

$$\mathsf{T}^{\mathcal{M}}_s = \bigcup_{\alpha \geq 0} \mathsf{T}^{(\alpha,\mathcal{M})}_s.$$

Moreover by definition of $\mathcal{L}_L$ there is a production of the form:

$$\frac{}{\mathsf{T}_s c} \in \Phi.$$

Hence by definition of the sets T^{α}_s we have $c^{\mathcal{M}} \in \mathsf{T}^{(1,\mathcal{M})}_s$ and therefore $c^{\mathcal{M}} \in \mathsf{T}^{\mathcal{M}}_s$. For the induction step suppose that t is of the form $f(t_1, \ldots, t_n)$ where $f : s_1 \to \cdots \to s_n \to s \in \Sigma$ and $t_i \in Gnd(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s_i))$ for all $i = 1, \ldots, n$. By the induction hypothesis we have $t_i \in \mathsf{T}^{\mathcal{M}}_s$, and hence there exists an ordinal α_i such that $t_i^{\mathcal{M}} \in \mathsf{T}^{(\alpha_i,\mathcal{M})}_s$, for all $i = 1, \ldots, n$. Moreover by the definition of $\mathcal{L}_L$ there exists a production of the form:

$$\frac{\mathsf{T}_{s_1} x_1 \ldots \mathsf{T}_{s_n} x_n}{\mathsf{T}_s f(x_1, \ldots, x_n)} \in \Phi.$$

Let ρ be any valuation such that $\rho(x_i) = t_i^{\mathcal{M}}$. Now by definition of the sets $\mathsf{T}^{(\alpha,\mathcal{M})}_s$ we obtain that:

$$\begin{aligned}[\![f(x_1, \ldots, x_n)]\!]^{\mathcal{M}}_{\rho} &= f^{\mathcal{M}}(\rho(x_1), \ldots, \rho(x_n)) = f^{\mathcal{M}}(t_1^{\mathcal{M}}, \ldots, t_n^{\mathcal{M}}) \\ &= f(t_1, \ldots, t_n)^{\mathcal{M}} \in \mathsf{T}^{(1+\max_{i=1}^{n}\{\alpha_i\},\mathcal{M})}_s.\end{aligned}$$

□

Lemma 5.6. *Let $\mathcal{M}$ be a standard $\mathcal{L}_L$ structure. Then for every $s \in \{\iota, \omega\}$ and for all $d \in \mathsf{T}^{\mathcal{M}}_s$, there exists a term $t \in Gnd(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s))$ such that $t^{\mathcal{M}} = d$.*

Proof. By the argument given in [BS10] the following holds:

$$\mathsf{T}^{\mathcal{M}}_s = \bigcup_{\alpha \geq 0} = \mathsf{T}^{(\omega,\mathcal{M})}_s,$$

for all $s \in \{\iota, \omega\}$. Hence we can show the lemma by proceeding by induction on the ordinal α up to ω. For the base case $\alpha = 0$ we have $\mathsf{T}^{(0,\mathcal{M})}_s = \varnothing$. The claim holds trivially in this case. For the induction step suppose that the claim holds up to α and consider the case for $\alpha + 1$. Observe that it suffices to consider only the elements in $\mathsf{T}^{(\alpha+1,\mathcal{M})}_s \setminus \mathsf{T}^{(\alpha,\mathcal{M})}_s$ since $\mathsf{T}^{(\alpha+1,\mathcal{M})}_s \subseteq \mathsf{T}^{(\alpha,\mathcal{M})}_s$. Let d be any element in $\mathsf{T}^{(\alpha+1,\mathcal{M})}_s \setminus \mathsf{T}^{(\alpha,\mathcal{M})}_s$, then by the definition of the sets $\mathsf{T}^{(\alpha,\mathcal{M})}_s$ and the definition of the language $\mathcal{L}_L$ there exists in Φ a production of the form:

$$\frac{\mathsf{T}_{s_1} x_1 \ldots \mathsf{T}_{s_n} x_n}{\mathsf{T}_s f(x_1, \ldots, x_n)},$$

and a valuation ρ such that $[\![f(x_1,\ldots,x_n)]\!]^{\mathcal{M}}_{\rho} = d$ and $\rho(x_i) \in \mathsf{T}^{\alpha}_{s_i}$ for all $i = 1,\ldots,n$. By the induction hypothesis there exists for every $i = 1,\ldots,n$ a term $t_i \in \mathit{Gnd}(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s_i))$ such that $t_i^{\mathcal{M}} = \rho(x_i)$. But since $f(t_1,\ldots,t_n)^{\mathcal{M}} = f^{\mathcal{M}}(t_1^{\mathcal{M}},\ldots,t_n^{\mathcal{M}}) = f^{\mathcal{M}}(\rho(x_1),\ldots,\rho(x_n)) = [\![f(x_1,\ldots,x_n)]\!]^{\mathcal{M}}_{\rho}$ holds, the desired term is $f(t_1,\ldots,t_n) \in \mathit{Gnd}(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s))$. □

Lemma 5.7. *Let $\mathcal{M}$ be an $\mathcal{L}_L$ model of $\mathrm{T}^{\mathsf{inj}}$, then for every two terms $t_1, t_2 \in \mathit{Gnd}(\mathbb{T}^{\Sigma}_{\mathcal{X}}(\omega))$ if $t_1 \neq t_2$, then $t_1^{\mathcal{M}} \neq t_2^{\mathcal{M}}$.*

Proof. We proceed by induction on the structure of the term t_1. For the base case let $t_1 = \mathsf{0}$. Since $t_1 \neq t_2$ there exists a term $t_2' \in \mathit{Gnd}(\mathbb{T}^{\Sigma}_{\mathcal{X}}(\omega))$ such that $t_2 = s(t_2')$. Suppose that $t_1^{\mathcal{M}} = t_2^{\mathcal{M}}$. Then let ρ be any valuation with $\rho(x) = t_2'^{\mathcal{M}}$ and observe that $\mathcal{M} \not\models_{\rho} \mathsf{0} \neq \mathsf{s}(x)$, and hence $\mathcal{M} \not\models \mathrm{T}^{\mathsf{inj}}$. This is a contradiction. For the step case suppose that t_1 is of the form $s(t_1')$ for some term $t_1' \in \mathit{Gnd}(\mathbb{T}^{\Sigma}_{\mathcal{X}}(\omega))$ and assume that $t_1^{\mathcal{M}} = t_2^{\mathcal{M}}$. We must distinguish between two cases:

1. If $t_2 = \mathsf{0}$, then we let again ρ be any valuation such that $\rho(x) = t_1'^{\mathcal{M}}$. Then we have $\mathcal{M} \not\models \mathsf{0} \neq \mathsf{s}(x)$ and $\mathcal{M} \not\models \mathrm{T}^{\mathsf{inj}}$. This contradicts the assumptions!
2. If $t_2 \neq \mathsf{0}$, then there exists a term $t_2' \in \mathit{Gnd}(\mathbb{T}^{\Sigma}_{\mathcal{X}}(\omega))$ such that $t_2 = \mathsf{s}(t_2')$ and $t_1' \neq t_2'$. Since $\mathcal{M} \models \mathrm{T}^{\mathsf{inj}}$ we have for any valuation ρ that $\mathcal{M} \models_{\rho} \mathsf{s}(x) \neq \mathsf{s}(y) \vee x = y$ and hence $t_1'^{\mathcal{M}} = t_2'^{\mathcal{M}}$. But by the induction hypothesis we obtain that $t_1'^{\mathcal{M}} \neq t_2'^{\mathcal{M}}$. Contradiction!

□

Proof of Proposition 5.3. Let $(\mathcal{M}, \rho_\eta)$ be any model of $\mathsf{T}_\omega(\eta) \wedge \mathrm{T}^{\mathsf{inj}} \wedge \lceil S \rceil$. Then define the L interpretation $\mathcal{I}$ by $\mathcal{I} = (n, \equiv)$ where $n \in \mathbb{N}$ such that $\overline{n}^{\mathcal{M}} = \rho_\eta(\eta)$ and the relation $\equiv$ is given by:

$$t_1 \equiv t_2 \text{ if and only if } t_1^{\mathcal{M}} = t_2^{\mathcal{M}} \text{ for all } t_1, t_2 \in \mathit{Gnd}(\mathcal{T}^{\Sigma}_{\mathcal{X}}).$$

We will first show that $\mathcal{I}$ is a well-defined L interpretation. Since $\mathcal{M} \models_{\rho_\eta} \mathsf{T}_\omega(\eta)$, we have $\rho_\eta(\eta) \in \mathsf{T}^{\mathcal{M}}_{\omega}$. Hence by Lemma 5.6 there exists a term $t \in \mathit{Gnd}(\mathbb{T}^{\Sigma}_{\mathcal{X}}(\omega))$ such that $t^{\mathcal{M}} = \rho_\eta(\eta)$. Therefore there exists a number $n \in \mathbb{N}$ such that $\overline{n} = t$ i.e. $\overline{n}^{\mathcal{M}} = \rho_\eta(\eta)$. Hence n is defined. Furthermore by Lemma 5.7 the number n is uniquely determined. Consider now the relation $\equiv$, by its definition it is clear that this relation is well-defined. We now have to show that $\equiv$ is a congruence relation on ι-terms and ω-terms, afterwards we will show that $\equiv$ is the syntactic equality on ω-terms. To this end suppose that $\equiv$ is not a congruence relation. Then there exists an $s \in \{\iota, \omega\}$ for which one of the properties of congruence relations is violated:

- Suppose there is a term $t \in \mathit{Gnd}(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s))$ such that $t \not\equiv t$, which implies that $t^{\mathcal{M}} \neq t^{\mathcal{M}}$. This is impossible.
- Suppose that there exist terms $t_1, t_2 \in \mathit{Gnd}(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s))$ such that $t_1 \equiv t_2$ but $t_2 \not\equiv t_2$. This implies that $t_1^{\mathcal{M}} = t_2^{\mathcal{M}}$ but $t_2^{\mathcal{M}} \neq t_1^{\mathcal{M}}$. This contradicts the symmetry of the equality relation.

- Suppose that there exist terms $t_1, t_2, t_3 \in \mathit{Gnd}(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s))$ such that $t_1 \equiv t_2$ and $t_2 \equiv t_3$ but $t_1 \not\equiv t_3$. It follows that $t_1^{\mathcal{M}} = t_2^{\mathcal{M}} = t_3^{\mathcal{M}}$ but $t_1^{\mathcal{M}} \neq t_3^{\mathcal{M}}$. Contradiction!
- Suppose that there exists a function symbol $f : (s_1, \ldots, s_n, s) \in \Sigma$ and terms $t_i, u_i \in \mathit{Gnd}(\mathbb{T}^{\Sigma}_{\mathcal{X}}(s_i))$ for $i = 1, \ldots, n$ such that $t_i \equiv u_i$ for all $i = 1, \ldots, n$ but $f(t_1, \ldots, t_n) \not\equiv f(u_1, \ldots, u_n)$. From this it follows that $t_i^{\mathcal{M}} = u_i^{\mathcal{M}}$ for all $i = 1, \ldots, n$ but $f^{\mathcal{M}}(t_1^{\mathcal{M}}, \ldots, t_n^{\mathcal{M}}) \neq f^{\mathcal{M}}(u_1^{\mathcal{M}}, \ldots, u_n^{\mathcal{M}})$. This contradicts the assumption that $f^{\mathcal{M}}$ is a function.

Hence it is the case that $\equiv$ is a congruence relation for ι-terms and ω-terms. By Lemma 5.7 it is immediate that $\equiv$ is the syntactic equality on ω-terms. It remains to show that the interpretation $\mathcal{I}$ is a model of the clause set S. We proceed indirectly and assume that $\mathcal{I}$ is not a model of S. Then there exists a constraint clause $\mathcal{C} \in S$ and a ground substitution σ with $\mathit{dom}(\sigma) = \mathit{var}(\mathcal{C})$ such that:

$$\begin{aligned} \eta^{\mathcal{I}} = \overline{n} = t\sigma \text{ for all } \eta \simeq t \in \mathit{ctr}(\mathcal{C}),\\ u_1\sigma \not\equiv u_2\sigma \text{ for all } u_1 \simeq u_2 \in \mathit{cls}(\mathcal{C}),\\ v_1\sigma \equiv v_2\sigma \text{ for all } v_1 \not\simeq v_2 \in \mathit{cls}(\mathcal{C}). \end{aligned}$$

Consider the valuation $\rho = \rho_\eta[x \mapsto x\sigma^{\mathcal{M}} : x \in \mathit{var}(\mathcal{C})]$ and observe that by Lemma 5.5 we have $x\sigma^{\mathcal{M}} \in \mathsf{T}_s^{\mathcal{M}}$ for all $x_s \in \mathit{var}(\mathcal{C})$. Hence we have $\mathcal{M} \not\models_\rho \neg\mathsf{T}_s x$ for all $x_s \in \mathit{var}(\mathcal{C})$. Consider now arbitrary literals $\eta \simeq t \in \mathit{ctr}(\mathcal{C})$, $u_1 \simeq u_2 \in \mathit{cls}(\mathcal{C})$ and $v_1 \not\simeq v_2 \in \mathit{cls}(\mathcal{C})$. Then from $\overline{n} = t\sigma$ and by Lemma 5.4 it follows that $\overline{n}^{\mathcal{M}} = \rho_\eta(\eta) = \rho(\eta) = [\![t]\!]_\rho^{\mathcal{M}}$ and therefore $\mathcal{M} \not\models_\rho \eta \neq t$. Moreover $u_1\sigma \not\equiv u_2\sigma$ and $v_1\sigma \equiv v_2\sigma$ implies that $u_1\sigma^{\mathcal{M}} \neq u_2\sigma^{\mathcal{M}}$ and $v_1\sigma^{\mathcal{M}} = v_2\sigma^{\mathcal{M}}$. By Lemma 5.4 we obtain that $[\![u_1]\!]_\rho^{\mathcal{M}} \neq [\![u_2]\!]_\rho^{\mathcal{M}}$ and $[\![v_1]\!]_\rho^{\mathcal{M}} = [\![v_2]\!]_\rho^{\mathcal{M}}$. Hence $\mathcal{M} \not\models_\rho u_1 = u_2$ and $\mathcal{M} \not\models_\rho v_1 \neq v_2$. This means that every literal of the conjunct $\lceil\mathcal{C}\rceil$ of the formula $\lceil S\rceil$ is not satisfied by the structure $\mathcal{M}$ under the valuation ρ. But this means that $\mathcal{M}$ is not a model of $\lceil S\rceil$. Contradiction! □

Theorem 5.8. *Let S be an L-clause set. The clause set S is satisfiable if and only if the formula $\mathsf{T}_\omega\eta \wedge \mathrm{T}^{\mathsf{inj}} \wedge \lceil S\rceil$ is satisfiable with respect to standard interpretations.*

Proof. The proof follows immediately from Propositions 5.4 and 5.3. □

5.3.2 Truth Preservation

Lemma 5.9. *Let $\mathcal{I}$ be an L-interpretation. We have $t^{\mathcal{M}^{\mathcal{I}}} = [t]_{\stackrel{\cdot}{\equiv}}$ for all $t \in \mathit{Gnd}(\mathcal{T}^{\Sigma}_{\mathcal{X}})$.*

Proof. We proceed by induction on the structure of the term t. For the base case suppose that $t = c$ for some $c/0 \in \Sigma$. Then by definition of $\mathcal{M}^I$ we have $t^{\mathcal{M}^{\mathcal{I}}} = c^{\mathcal{M}^{\mathcal{I}}} = [c]_{\stackrel{\cdot}{\equiv}}$. For the step case let t be of the form $t = f(t_1, \ldots, t_n)$ for some $f/n \in \Sigma$. By the induction hypothesis we have $t_i^{\mathcal{M}^{\mathcal{I}}} = [t_i]_{\stackrel{\cdot}{\equiv}}$ for all $i = 1, \ldots, n$. By the definition of $\mathcal{M}^{\mathcal{I}}$ we then obtain the following:

$$\begin{aligned} t^{\mathcal{M}^{\mathcal{I}}} &= f(t_1, \ldots, t_n)^{\mathcal{M}^{\mathcal{I}}} = f^{\mathcal{M}^{\mathcal{I}}}(t_1^{\mathcal{M}^{\mathcal{I}}}, \ldots, t_n^{\mathcal{M}^{\mathcal{I}}})\\ &= f^{\mathcal{M}^{\mathcal{I}}}([t_1]_{\stackrel{\cdot}{\equiv}}, \ldots, [t_n]_{\stackrel{\cdot}{\equiv}}) = [f(t_1, \ldots, t_n)]_{\stackrel{\cdot}{\equiv}}. \end{aligned}$$

□

Lemma 5.10. *Let S be an L-clause set and let $\mathcal{I}$ be a L-interpretation. If $\mathcal{I} \not\models S$, then $\mathcal{M}^{\mathcal{I}} \not\models_{\{\eta \to \{\overline{\eta^{\mathcal{I}}}\}\}} \lceil S \rceil$.*

Proof. Assume $\mathcal{I} \not\models S$. Then there exists a constraint clause $\mathcal{C} \in S$ and a typed ground substitution σ such that for every $\eta \simeq t \in ctr(\mathcal{C})$ we have $\overline{\eta^{\mathcal{I}}} \equiv^{\mathcal{I}} t\sigma$ and $\mathcal{I} \not\models cls(\mathcal{C})\sigma$.

We need to show that there exists a valuation such that every literal of $\lceil S \rceil$ is falsified. To this end we define $\rho = \{x\sigma^{\mathcal{M}^{\mathcal{I}}} : x \in dom(\sigma)\} \cup \{\eta \mapsto \{\overline{\eta}^{\mathcal{I}}\}\}$.

Let $x_s \in var(\mathcal{C})$, then by Lemma 5.4 we have $x\sigma^{\mathcal{M}^{\mathcal{I}}} \in \mathsf{T}_s^{\mathcal{M}^{\mathcal{I}}}$. By Lemma 5.9 we obtain $[\![x]\!]_\rho^{\mathcal{M}^{\mathcal{I}}} = x\sigma^{\mathcal{M}^{\mathcal{I}}}$ and therefore $\mathcal{M}^{\mathcal{I}} \not\models_\rho \neg \mathsf{T}_s x$.

Let now $\eta \simeq t$ be any constraint literal of $\mathcal{C}$. By the assumptions we have $\overline{\eta^{\mathcal{I}}} \equiv^{\mathcal{I}} t\sigma$ and by the definition of the relation $\check{\equiv}$ we have $\overline{\eta^{\mathcal{I}}} \check{\equiv} t\sigma$ i.e. $[\overline{\eta^{\mathcal{I}}}]_{\check{\equiv}} = [t\sigma]_{\check{\equiv}}$. By Lemma 5.9 we have $[\![t]\!]_\rho^{\mathcal{M}^{\mathcal{I}}} = [t\sigma]_{\check{\equiv}}$ and thus we obtain $\mathcal{M}^{\mathcal{I}} \not\models_\rho \eta \neq t$.

Let $u_1 \simeq u_2$ be any positive literal in $cls(\mathcal{C})$. Then we have by the assumptions $u_1\sigma \not\equiv^{\mathcal{I}} u_2\sigma$ and thus $u_1\sigma \check{\not\equiv} u_2\sigma$. By Lemmata 5.9 and 5.4 we obtain

$$[\![u_1]\!]_\rho^{\mathcal{M}^{\mathcal{I}}} = [u_1\sigma]_{\check{\equiv}} \text{ and } [\![u_2]\!]_\rho^{\mathcal{M}^{\mathcal{I}}} = [u_2\sigma]_{\check{\equiv}}.$$

Therefore we have $\mathcal{M}^{\mathcal{I}} \not\models_\rho u_1 = u_2$. Analogously we obtain $\mathcal{M}^{\mathcal{I}} \not\models v_1 \neq v_2$ for every negative literal $v_1 \not\simeq v_2 \in cls(\mathcal{C})$.

Finally, by the definition of the formula $\lceil \mathcal{C} \rceil$ we have $\mathcal{M} \not\models_\rho \overline{\lceil \mathcal{C} \rceil}$ and therefore $\mathcal{M}^{\mathcal{I}} \not\models \lceil S \rceil$. □

Proposition 5.11. *Let S_1, S_2 be L-clause sets. If $\lceil S_1 \rceil \models \lceil S_2 \rceil$, then we have $S_1 \models S_2$.*

Proof. We proceed indirectly by assuming $\lceil S_1 \rceil \models \lceil S_2 \rceil$ but $S_1 \not\models S_2$. Then there exists an L-interpretation $\mathcal{I}$ such that $\mathcal{I} \models S_1$ and $\mathcal{I} \not\models S_2$. By Lemmas 5.1 and 5.10 we have $\mathcal{M}^{\mathcal{I}} \models_{\{\eta \mapsto \{\overline{\eta^{\mathcal{I}}}\}\}} \lceil S_1 \rceil$ and $\mathcal{M}^{\mathcal{I}} \not\models_{\{\eta \mapsto \{\overline{\eta^{\mathcal{I}}}\}\}} \lceil S_2 \rceil$. Contradiction! □

Conjecture 1. *Let S_1, S_2 be L-clause sets. If $S_1 \models S_2$, then $\mathrm{T}^{\mathsf{inj}}, \mathsf{T}_\omega \eta, \lceil S_1 \rceil \models \lceil S_2 \rceil$.*

CHAPTER 6

Translation of the n-Clause Calculus to LKID

This chapter describes the derivation of induction invariants that summarize the inductive arguments captured by the n-clause calculus. We proceed in five major steps. In Section 6.1 we show that clause normalization can be simulated in the induction-free system LKID-I. Section 6.2 shows that superposition deductions translate to induction-free proofs, i.e. proofs in the subsystem LKID-I. In Section 6.3 we will show that the inductive cycles of the n-clause calculus can be formulated in the cyclic proof system CLKID^{ω}. Section 6.4 shows the translation of a simple type of cyclic proof to a corresponding type of inductive proof. And finally, in Section 6.5 we put the obtained results together to obtain an LKID representation of n-clause refutations.

In the following we will often have to deal with the type assertions introduced by the translation described in Definition 5.3. The two lemmas below will help us to deal with these type assertions. Informally, Lemma 6.1 states that the type assertion for a term t of sort s can be derived in LKID-I if every free variable of t has a suitable type assertion.

Lemma 6.1. *Let $x_1 : s_1, \ldots, x_n : s_n$ be L-variables and $t \in \mathbb{T}^{\Sigma}_{\{x_1,\ldots,x_n\}}(s)$. Then the sequent*

$$\Gamma, \mathsf{T}_{s_1} x_1, \ldots, \mathsf{T}_{s_n} x_n \Rightarrow \mathsf{T}_s t, \Delta$$

is provable in LKID-I.

Proof. We proceed by induction on the structure of the term t. If $t = y$ where y is a variable, then $y : s \in \{x_1 : s_1, \ldots, x_n : s\}$ and therefore $\mathsf{T}_s t \in \{\mathsf{T}_{s_1} x_1, \ldots, \mathsf{T}_{s_n} x_n\}$. Hence the sequent is a logical axiom. For the induction step suppose that $t = f(t_1, \ldots, t_m)$, where $f : s'_1 \to \ldots s'_m \to s \in \Sigma$ and $t_i \in \mathbb{T}^{\Sigma}_{\{x_1,\ldots,x_n\}}(s'_i)$ for $1 \leq i \leq m$. Then by the induction hypothesis there exists an LKID-I proof π_i of the sequent $\Gamma, \mathsf{T}_{s_1} x_1, \ldots, \mathsf{T}_{s_n} x_n \Rightarrow \mathsf{T}_{s'_i} t_i$ for $i = 1, \ldots, m$. Therefore the following is an LKID-I proof of the original sequent.

$$\frac{\begin{array}{ccc}\vdots\,\pi_1 & & \vdots\,\pi_m\\ \Gamma, \mathsf{T}_{s_1}x_1, \ldots, \mathsf{T}_{s_n}x_n \Rightarrow \mathsf{T}_{s'_1}t_1, \Delta & \ldots & \Gamma, \mathsf{T}_{s_1}x_1, \ldots, \mathsf{T}_{s_n}x_n \Rightarrow \mathsf{T}_{s'_m}t_m, \Delta\end{array}}{\Gamma, \mathsf{T}_{s_1}x_1, \ldots, \mathsf{T}_{s_n}x_n \Rightarrow \mathsf{T}_s f(t_1, \ldots, t_m), \Delta}\,(\mathsf{T}_s\mathrm{R})$$

□

Lemma 6.2. *Let $\mathcal{C}, \mathcal{D}$ be L-constraint clauses and θ a well-typed substitution. If for all $x : s \in var(\mathcal{C})$ we have $var(x\theta) \subseteq var(\mathcal{D})$, then the sequent*

$$\lceil \mathcal{D} \rceil_{\mathsf{type}} \Rightarrow \lceil \mathcal{C} \rceil_{\mathsf{type}}\theta$$

is provable in LKID-I.

Proof. Let $x : s \in var(\mathcal{C})$, then $var(x\theta) = \{y_1 : s_1, \ldots, y_n : s_n\} \subseteq var(\mathcal{D})$ and by definition the formula $\lceil \mathcal{D} \rceil_{\mathsf{type}}$ contains conjuncts $\mathsf{T}_{s_1}y_1, \ldots, \mathsf{T}_{s_n}y_n$. By Lemma 6.1 the sequent $\lceil \mathcal{D} \rceil_{\mathsf{type}} \Rightarrow \mathsf{T}_{s_i}x\theta$ is provable in LKID-I. Since $\lceil \mathcal{C} \rceil\theta = \bigwedge_{x:s \in var(\mathcal{C})} \mathsf{T}_s x\theta$ the sequent $\lceil \mathcal{D} \rceil_{\mathsf{type}} \Rightarrow \lceil \mathcal{C} \rceil_{\mathsf{type}}\theta$ is provable in LKID-I. □

6.1 Clause Normalization

We have seen in Chapter 3 that constraint clauses are normalizable in the sense that every n-clause with a non-empty constraint part is logically equivalent to some constraint clause having exactly one atom in its constraint part. Moreover, this normalization is crucial for the detection of some cycles. Thus, we need to simulate clause normalization on the translation of constraint clauses.

Lemma 6.3. *The sequent* $\mathsf{inj^s}, \mathsf{s}^m\alpha = \mathsf{s}^m\beta \Rightarrow \alpha = \beta$ *is provable in* LKID-I *for all $m \in \mathbb{N}$ with $m > 0$.*

Proof. The proof is by weak induction on the number m. If $m = 1$, then then we obtain the desired proof by simple applications of ∀R and →L. For the induction step, suppose that there exists a proof π_m of the sequent

$$\mathsf{inj^s}, s^m\alpha = s^m\beta \Rightarrow \alpha = \beta.$$

Then the following is the desired LKID-I proof

$$\dfrac{\dfrac{}{s^{m+1}\alpha = s^{m+1}\beta \Rightarrow s^{m+1}\alpha = s^{m+1}\beta}\,(\mathrm{Ax}) \qquad \begin{array}{c}\vdots\,\pi_m\\ \mathsf{inj^s}, \mathsf{s}^m\alpha = \mathsf{s}^m\beta \Rightarrow \alpha = \beta\end{array}}{\dfrac{\mathsf{inj^s}, \mathsf{ss}^m\alpha = \mathsf{ss}^m\beta \rightarrow \mathsf{s}^m\alpha = \mathsf{s}^m\beta, s^{m+1}\alpha = s^{m+1}\beta \Rightarrow \alpha = \beta}{\mathsf{inj^s}, s^{m+1}\alpha = s^{m+1}\beta \Rightarrow \alpha = \beta.}\,(\forall\mathrm{L})}\,(\rightarrow\mathrm{L})$$

□

Lemma 6.4. *Let $\mathcal{C} = [C \mid \eta \simeq t_1, \ldots, \eta \simeq t_n]$ be an n-clause. If $t_1, \ldots, t_n$ are not unifiable, then the sequent* $\mathrm{T}^{\mathsf{inj}} \Rightarrow \lceil \mathcal{C} \rceil$ *is provable in* LKID-I.

Proof. If the terms $t_1, \ldots, t_n$ are not unifiable, there exist $i, j \in \{1, \ldots, n\}$ with $i \neq j$ such that t_i and t_j are not unifiable. Without loss of generality we have $t_i = \mathsf{s}^n t$ with or $t \in \{x, \mathsf{0}\}$, and $t_j = \overline{m}$ with $n > m$ i.e. $n = m + k$ for some $k > 0$. By Lemma 6.3, there exists an LKID-I proof π of the sequent $\mathrm{T}^{\mathsf{inj}}, \mathsf{s}^m\mathsf{s}^k t = \mathsf{s}^m\mathsf{0} \Rightarrow \mathsf{s}^k t = \mathsf{0}$. Therefore, we prove the sequent as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{s}^k t = \mathsf{0} \Rightarrow \mathsf{s}^k t = \mathsf{0}}\,(\mathrm{Ax})}{\mathsf{s}^k t \neq \mathsf{0}, \mathsf{s}^k t = \mathsf{0} \Rightarrow}\,(\neg\mathrm{L})}{\mathrm{T}^{\mathsf{inj}}, \mathsf{s}^k t = \mathsf{0} \Rightarrow}\,(\wedge\mathrm{L}, \forall\mathrm{L})}{\mathrm{T}^{\mathsf{inj}}, \mathsf{s}^m\mathsf{s}^k t = \mathsf{s}^m\mathsf{0} \Rightarrow}\,(\mathrm{Cut})}{\mathrm{T}^{\mathsf{inj}}, \eta = \mathsf{s}^n t, \eta = \overline{m} \Rightarrow}\,(=\mathrm{L})}{\mathrm{T}^{\mathsf{inj}} \Rightarrow \overline{\lceil \mathcal{C} \rceil}}\,(\rightarrow\mathrm{R}, \mathrm{Wk})}{\mathrm{T}^{\mathsf{inj}} \Rightarrow \lceil \mathcal{C} \rceil}\,(\forall\mathrm{R})$$

π

□

Lemma 6.5. *Let $\mathcal{C} = [C \mid \eta \simeq t_1, \ldots, \eta \simeq t_n]$ be a constraint clause. If the terms $t_1, \ldots, t_n$ are unifiable with m.g.u. σ, the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil \mathcal{C} \rceil \Rightarrow \lceil \mathcal{C}' \rceil$ is provable in* LKID-I*, where $\mathcal{C}' = [C\sigma \mid \eta \simeq t_1\sigma]$.*

Proof. We decompose the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil \mathcal{C} \rceil \Rightarrow \lceil \mathcal{C}' \rceil$ as follows.

$$\dfrac{\dfrac{\lceil \mathcal{C}' \rceil_{\mathsf{type}} \Rightarrow \lceil \mathcal{C} \rceil_{\mathsf{type}}\sigma \qquad \lceil \mathcal{C}' \rceil_{\mathsf{constraint}} \Rightarrow \lceil \mathcal{C} \rceil_{\mathsf{constraint}}\sigma \qquad \lceil \mathcal{C} \rceil_{\mathsf{clause}}\sigma \Rightarrow \lceil \mathcal{C}' \rceil_{\mathsf{clause}}}{\mathrm{T}^{\mathsf{inj}}, \overline{\lceil \mathcal{C} \rceil}\sigma \Rightarrow \overline{\lceil \mathcal{C}' \rceil}}\,(\rightarrow\mathrm{R}, \rightarrow\mathrm{L})}{\mathrm{T}^{\mathsf{inj}}, \lceil \mathcal{C} \rceil \Rightarrow \lceil \mathcal{C}' \rceil}\,(\forall\mathrm{R}, \forall\mathrm{L})$$

For the sequent $\lceil \mathcal{C}' \rceil_{\mathsf{type}} \Rightarrow \lceil \mathcal{C} \rceil_{\mathsf{type}}\sigma$ we obtain an LKID-I proof by Lemma 6.2. Observe that the sequent $\lceil \mathcal{C}' \rceil_{\mathsf{constraint}} \Rightarrow \lceil \mathcal{C} \rceil_{\mathsf{constraint}}\sigma$ is of the form

$$\eta = t_1\sigma \Rightarrow \eta = t_1\sigma \wedge \ldots, \eta = t_n\sigma.$$

Since $t_1\sigma = t_2\sigma = \cdots = t_n\sigma$, this sequent is proved by straightforward $\wedge$R inferences. For the sequent $\lceil \mathcal{C} \rceil_{\mathsf{clause}}\sigma \Rightarrow \lceil \mathcal{C}' \rceil_{\mathsf{clause}}$ observe that $\lceil \mathcal{C} \rceil_{\mathsf{clause}}\sigma = \lceil \mathcal{C}' \rceil_{\mathsf{clause}}$. Hence, this sequent is an axiom. □

Lemma 6.6. *Let $\{t_1, \ldots, t_n\}$ be a unifiable set of ω-terms with m.g.u. σ. Then there exists $t \in \{t_1, \ldots, t_n\}$ such that $t\sigma = t$.*

Proof. If there are ground terms among $\{t_1, \ldots, t_n\}$, then any maximum ground term satisfies the desired properties. Otherwise, there exists in $\{t_1, \ldots, t_n\}$ a maximum term $\mathsf{s}^m x$ with $x\sigma = x$. □

Lemma 6.7. *Let $\mathcal{C} = [C \mid \eta \simeq t_1, \ldots, \eta \simeq t_n]$ be a constraint clause. If the terms $t_1, \ldots, t_n$ are unifiable with m.g.u. σ, then the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil \mathcal{C}' \rceil \Rightarrow \lceil \mathcal{C} \rceil$, with $\mathcal{C}' = [C\sigma \mid \eta \simeq t_1\sigma]$ is provable in* LKID-I*.*

Proof. By a sequence of straightforward first-order inferences, we decompose the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil \mathcal{C}' \rceil \Rightarrow \lceil \mathcal{C} \rceil$ into the sequents below:

$$\lceil \mathcal{C} \rceil_{\mathsf{type}} \Rightarrow \lceil \mathcal{C}' \rceil_{\mathsf{type}},$$
$$\lceil \mathcal{C} \rceil_{\mathsf{constraint}} \Rightarrow \lceil \mathcal{C}' \rceil_{\mathsf{constraint}},$$
$$\mathrm{T}^{\mathsf{inj}}, \eta = t_1, \ldots, \eta = t_n, \lceil C\sigma \rceil \Rightarrow \lceil C \rceil.$$

Since σ is an m.g.u. of $t_1, \ldots, t_n$, we have $var\,\mathcal{C} \supseteq var(\mathcal{C}')$. Therefore, the sequent $\lceil \mathcal{C} \rceil_{\mathsf{type}} \Rightarrow \lceil \mathcal{C}' \rceil_{\mathsf{type}}$ can be proved by a sequence of $\wedge$R and $\wedge$L inferences.

Observe that the sequent $\lceil \mathcal{C} \rceil_{\mathsf{constraint}} \Rightarrow \lceil \mathcal{C}' \rceil_{\mathsf{constraint}}$ is of the form

$$\eta = t_1 \wedge \cdots \wedge \eta = t_n \Rightarrow \eta = t_1\sigma.$$

By Lemma 6.6, we have $t_m\sigma = t_m = t_1\sigma$ for some $m \in \{1, \ldots, n\}$. Hence, the sequent $\lceil \mathcal{C} \rceil_{\mathsf{constraint}} \Rightarrow \lceil \mathcal{C}' \rceil_{\mathsf{constraint}}$ is clearly provable in LKID-I.

Now consider the sequent $\mathrm{T}^{\mathsf{inj}}, \eta = t_1, \ldots, \eta = t_n, \lceil \mathcal{C}' \rceil_{\mathsf{clause}} \Rightarrow \lceil \mathcal{C} \rceil_{\mathsf{clause}}$. Let again t_m be the term with $t_m\sigma = t_m$ obtained by Lemma 6.6. Then by applying several =L inferences we obtain the sequent

$$\mathrm{T}^{\mathsf{inj}}, t_m = t_1, \ldots, t_m = t_n, \lceil C\sigma \rceil \Rightarrow \lceil C \rceil.$$

For $i = 1, \ldots, n$, the term t_i is of the form $\mathsf{s}^{k_i}c_i$ with $c_i \in \{0, x_i\}$, where x_i is a variable. Moreover, since $t_1, \ldots, t_n$ are unifiable, we have $k_m \geq k_i$, for all $i \in \{1, \ldots, n\}$. Hence each atom $t_m = t_i$ is of the form $\mathsf{s}^{k_i}\mathsf{s}^{m_i}c_m = \mathsf{s}^{k_i}c_i$, with $m_i = k_m - k_i$. Therefore, by repeated applications of Cut and Lemma 6.3 we obtain the sequent

$$\mathrm{T}^{\mathsf{inj}}, \mathsf{s}^{m_1}c_m = c_1, \ldots, \mathsf{s}^{m_n}c_m = c_n, \lceil C\sigma \rceil \Rightarrow \lceil C \rceil.$$

Since $t_1, \ldots, t_n$ is unifiable, each atom $\mathsf{s}^{m_i} = c_i$ is either of the form $\mathsf{0} = \mathsf{0}$ or $\mathsf{s}^{m_i}x_m = x_i$. Hence, these equational atoms describe the unifier σ. Thus, we finish the proof by using a sequence of suitable =L inferences. □

Proposition 6.8. *Let $\mathcal{C} = [C \mid \eta \simeq t_1, \ldots, \eta \simeq t_n]$ be a constraint clause. If $t_1, \ldots, t_n$ are unifiable, then the sequent $\mathrm{T}^{\mathsf{inj}} \Rightarrow \lceil \mathcal{C} \rceil \leftrightarrow \lceil \mathcal{C}' \rceil$, with $\mathcal{C}' = [C\sigma \mid \eta \simeq t_1\sigma]$, is provable in* LKID-I. *Otherwise, the sequent $\mathrm{T}^{\mathsf{inj}} \Rightarrow \lceil \mathcal{C} \rceil$ is provable in* LKID-I.

Proof. An immediate consequence of Lemmas 6.4, 6.5, and 6.7. □

6.2 Superposition Deductions

In the following we will translate superposition deductions to proofs in the system LKID-I. This is possible because the superposition deductions of the n-clause calculus capture first-order information only. The inference rules used by the superposition calculus can be simulated by LKID-I. The following Proposition is the main result of this section.

Proposition 6.9. *Let S be an L-clause set and let $\mathcal{C}$ be an L-constraint clause. If there exists a superposition deduction of $\mathcal{C}$ from S, then the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S \rceil \Rightarrow \lceil \mathcal{C} \rceil$ is provable in* LKID-I.

We prove the above proposition by induction on the length of the superposition deduction and by case distinction on the last inference rule. We shall first show how each type of inference is translated before we proceed to the proof of Proposition 6.9.

6.2.1 Inferences Rules

The three lemmas below are proved in a similar way: the end-sequent is decomposed and the quantifiers are instantiated according to the m.g.u. that is used by the rule instance; a number of more or less trivial cases needs to be considered and finally one branch of the proof justifies the actual inference. There is one detail that needs to be kept in mind. Sometimes the conclusion of the inference contains more variables than the substitution instances of the premises. This is relevant for us since the translation relativizes variables to the domain of their sort. But in this case, by the way we substitute, there are terms that contain variables without type assertions. To avoid such a situation, we apply an arbitrary ground substitution so that there are no such variables.

Lemma 6.10. *Let $\mathcal{C}_1 = [C_1 \vee t \bowtie s \mid X_1]$ and $\mathcal{C}_2 = [C_2 \vee u \simeq v \mid X_2]$ be variable-disjoint L-constraint clauses with $\bowtie \in \{\simeq, \not\simeq\}$, p a position such that u and $t|_p$ are unifiable with m.g.u. σ. Then the sequent*

$$\lceil \mathcal{C}_1, \mathcal{C}_2 \rceil \Rightarrow \lceil \sup_{t \bowtie s, u \simeq v, p}(\mathcal{C}_1, \mathcal{C}_2) \rceil,$$

is provable in LKID-I.

Proof. We start by letting

$$\mathcal{D} = \sup_{t \bowtie s, u \simeq v, p}(\mathcal{C}_1, \mathcal{C}_2),$$
$$\mathcal{Y} = var\{x\sigma : x \in var\{\mathcal{C}_1, \mathcal{C}_2\}\} \setminus var(\mathcal{D}).$$

Now observe that none of the variables in $\mathcal{Y}$ occurs in $\mathcal{C}_1\sigma$ and $\mathcal{C}_2\sigma$ outside of $t|_p\sigma$ and $u\sigma$. Since the signature of the language contains at least one individual constant for each sort there exists a ground substitution μ for variables $\mathcal{Y}$. Then we have

$$\mathcal{C}_1\sigma\mu = [C_1\sigma \vee t\sigma\mu \bowtie s\sigma \mid X_1\sigma] \text{ and } \mathcal{C}_2\sigma\mu = [C_2\sigma \vee u\sigma\mu \simeq v\sigma \mid X_2\sigma],$$

where $t\sigma\mu = t\sigma[t|_p\sigma\mu]_p$. Therefore we also have

$$var\{\mathcal{C}_1\sigma\mu, \mathcal{C}_2\sigma\mu\} = var(\mathcal{D}).$$

We decompose the sequent $\lceil \mathcal{C}_1 \rceil, \lceil \mathcal{C}_2 \rceil \Rightarrow \lceil \mathcal{D} \rceil$ as follows:

$$\dfrac{\dfrac{\dfrac{S_1 \quad S_2 \quad S_3 \quad S_4 \quad S_5}{\lceil\mathcal{D}\rceil_{\mathsf{type}}\lceil\mathcal{D}\rceil_{\mathsf{constraint}}, \overline{\lceil\mathcal{C}_1\rceil}\sigma\mu, \overline{\lceil\mathcal{C}_2\rceil}\sigma\mu \Rightarrow \lceil\mathcal{D}\rceil_{\mathsf{clause}}}\,(\to\mathrm{L}, \wedge\mathrm{R})}{\overline{\lceil\mathcal{C}_1\rceil}\sigma\mu, \overline{\lceil\mathcal{C}_2\rceil}\sigma\mu \Rightarrow \overline{\lceil\mathcal{D}\rceil}}\,(\to\mathrm{R}, \wedge\mathrm{L})}{\lceil\mathcal{C}_1\rceil, \lceil\mathcal{C}_2\rceil \Rightarrow \lceil\mathcal{D}\rceil}\,(\forall\mathrm{R}, \forall\mathrm{L})$$

where the sequents $S_1, \ldots, S_5$ are given by

$$\begin{aligned}
S_1 &= \lceil\mathcal{D}\rceil_{\mathsf{type}} \Rightarrow \lceil\mathcal{C}_1\rceil_{\mathsf{type}}\sigma\mu,\\
S_2 &= \lceil\mathcal{D}\rceil_{\mathsf{type}} \Rightarrow \lceil\mathcal{C}_2\rceil_{\mathsf{type}}\sigma\mu,\\
S_3 &= \lceil\mathcal{D}\rceil_{\mathsf{constraint}} \Rightarrow \lceil\mathcal{C}_1\rceil_{\mathsf{constraint}}\sigma\mu,\\
S_4 &= \lceil\mathcal{D}\rceil_{\mathsf{constraint}} \Rightarrow \lceil\mathcal{C}_2\rceil_{\mathsf{constraint}}\sigma\mu,\\
S_5 &= \lceil\mathcal{C}_1\rceil_{\mathsf{clause}}\sigma\mu, \lceil\mathcal{C}_2\rceil_{\mathsf{clause}}\sigma\mu \Rightarrow \lceil\mathcal{D}\rceil_{\mathsf{clause}}.
\end{aligned}$$

In order to obtain an LKID-I proof of the sequent $\lceil\mathcal{C}_1\rceil, \lceil\mathcal{C}_2\rceil \Rightarrow \mathcal{D}$ we need to show that each of the sequents $S_1, \ldots, S_5$ admits an LKID-I proof.

By Lemma 6.2 the sequents S_1 and S_2 are provable in LKID-I. Consider now the sequent S_3 and observe that the definition of $\lceil\cdot\rceil$ implies $\lceil\mathcal{C}_1\rceil_{\mathsf{constraint}}\sigma\mu = \lceil\mathcal{C}\sigma\mu\rceil_{\mathsf{constraint}}$. Therefore every conjunct of $\lceil\mathcal{C}_1\rceil_{\mathsf{constraint}}\sigma\mu$ is also a conjunct of $\lceil\mathcal{D}\rceil_{\mathsf{constraint}}$. Hence the sequent S_3 is clearly provable in LKID-I. Proceed similarly for the sequent S_4.

Consider now the sequent S_5. Without loss of generality this sequent is of the form

$$(t \bowtie s)\sigma\mu \vee l_1 \vee \cdots \vee l_n, (u = v)\sigma\mu \vee l_{n+1} \vee \cdots \vee l_m \Rightarrow l_1 \vee \cdots \vee l_m \vee t_\sigma[v\sigma]_p \bowtie s\sigma.$$

Hence we can prove the sequent S_5 as follows.

$$\dfrac{\dfrac{}{l_1 \Rightarrow l_1}\,(\mathrm{Ax}) \quad \ldots \quad \dfrac{}{l_m \Rightarrow l_m}\,(\mathrm{Ax}) \quad \dfrac{\dfrac{}{t\sigma[v\sigma]p \bowtie s\sigma \Rightarrow t\sigma[v\sigma]p \bowtie s\sigma}\,(=\mathrm{R})}{t\sigma[t|_p\sigma\mu]_p \bowtie s\sigma, u\sigma\mu = v\sigma \Rightarrow t\sigma[v\sigma]_p \bowtie s\sigma}\,(=\mathrm{L})}{S_5}\,(\vee\mathrm{L}, \vee\mathrm{R}, \mathrm{Wk})$$

□

Lemma 6.11. *Let $\mathcal{C} = [C \vee t \not\approx s \mid X]$ be an L-constraint clause such that t and s are unifiable with m.g.u. σ. Then the sequent*

$$\lceil\mathcal{C}\rceil \Rightarrow \lceil\mathrm{refl}_{t\not\approx s}(\mathcal{C})\rceil,$$

is provable in LKID-I.

Proof. We have to consider the case that $var(\mathcal{C}\sigma) \supset var(\mathrm{refl}_{t\not\approx s}(\mathcal{C}))$. Let $\mathcal{Y} = var(\mathcal{C}\sigma) \setminus var(\mathrm{refl}_{t\not\approx s}(\mathcal{C}))$ and observe that no variable in $\mathcal{Y}$ occurs in $\mathcal{C}\sigma$ outside of the literal $(t \not\approx s)\sigma$. Let μ be any well-typed ground substitution with $dom(\mu) = \mathcal{Y}$. Then we have $\mathcal{C}\sigma\mu = [\mathcal{C}\sigma \vee t\sigma\mu \not\approx s\sigma\mu \mid X\sigma]$ and $t\sigma\mu = s\sigma\mu$ since $t\sigma = s\sigma$. Moreover we have $var(\mathcal{C}\sigma\mu) = var(\mathrm{refl}_{t\not\approx s}(\mathcal{C}))$. We start by decomposing the sequent $\lceil\mathcal{C}\rceil \Rightarrow \lceil\mathcal{D}\rceil$ as follows:

$$\dfrac{\dfrac{\dfrac{S_1 \qquad S_2 \qquad S_3}{\lceil\mathcal{D}\rceil_{\mathsf{type}}, \lceil\mathcal{D}\rceil_{\mathsf{constraint}}, \overline{\lceil\mathcal{C}\rceil}\sigma\mu \Rightarrow \lceil\mathcal{D}\rceil_{\mathsf{clause}}}\,(\to\mathrm{L}, \wedge\mathrm{R})}{\overline{\lceil\mathcal{C}\rceil}\sigma\mu \Rightarrow \overline{\lceil\mathcal{D}\rceil}}\,(\to\mathrm{R}, \wedge\mathrm{L})}{\lceil\mathcal{C}\rceil \Rightarrow \lceil\mathcal{D}\rceil}\,(\forall\mathrm{R}, \forall\mathrm{L})$$

where the sequents S_1, S_2 and S_3 are given by

$$\begin{aligned} S_1 &= \lceil\mathcal{D}\rceil_{\mathsf{type}} \Rightarrow \lceil\mathcal{C}\rceil_{\mathsf{type}}\sigma\mu, \\ S_2 &= \lceil\mathcal{D}\rceil_{\mathsf{constraint}} \Rightarrow \lceil\mathcal{C}\rceil_{\mathsf{constraint}}\sigma\mu, \\ S_3 &= \lceil\mathcal{C}\rceil_{\mathsf{clause}}\sigma\mu \Rightarrow \mathcal{D}_{\mathsf{clause}}. \end{aligned}$$

It remains to show that the sequents S_1, S_2 and S_3 are provable in LKID-I. By Lemma 6.2 sequent S_1 is provable in LKID-I. Now consider the sequent S_2 and observe that $\lceil\mathcal{C}\rceil_{\mathsf{constraint}}\sigma\mu = \lceil\mathcal{C}\sigma\mu\rceil_{\mathsf{constraint}}$ and $ctr(\mathcal{D}) = ctr(\mathcal{C}\sigma\mu)$. By the construction of $\lceil\cdot\rceil$ the sequent S_2 is clearly provable in LKID-I. Since $\lceil\mathcal{C}\rceil_{\mathsf{clause}}\sigma\mu = \lceil\mathcal{C}\sigma\mu\rceil_{\mathsf{clause}}$ and $t\sigma\mu = s\sigma\mu$ the sequent S_3 is without loss of generality of the form

$$t\sigma\mu \not\simeq t\sigma\mu \vee l_1 \vee \cdots \vee l_n \Rightarrow l_1 \vee \cdots \vee l_n.$$

Therefore we can prove it as follows

$$\dfrac{\dfrac{}{l_1 \Rightarrow l_1}\,(\mathrm{Ax}) \quad \ldots \quad \dfrac{}{l_n \Rightarrow l_n}\,(\mathrm{Ax}) \quad \dfrac{\dfrac{}{\Rightarrow t\sigma\mu = t\sigma\mu}\,(\mathrm{Ax})}{t\sigma\mu \not\simeq t\sigma\mu \Rightarrow}\,(\neg\mathrm{L})}{S_3}\,(\vee\mathrm{L}, \vee\mathrm{R}, \mathrm{Wk})$$

□

Lemma 6.12. *Let $\mathcal{C} = [C \vee t \simeq s \vee u \simeq v \mid X]$ be an L-constraint clause such that t and u are unifiable with m.g.u. σ. Then the sequent*

$$\lceil\mathcal{C}\rceil \Rightarrow \lceil\mathrm{fact}_{t\simeq s, u\simeq v}(\mathcal{C})\rceil,$$

is provable in LKID-I.

Proof. We start by letting $\mathcal{D} = \mathrm{fact}_{t\simeq s, u\simeq v}(\mathcal{C})$. Observe that $var(\mathcal{C}\sigma) \setminus var(\mathcal{D}) = \varnothing$. Now we decompose the sequent $\lceil\mathcal{C}\rceil \Rightarrow \lceil\mathcal{D}\rceil$ as follows:

$$\dfrac{\dfrac{\dfrac{S_1 \qquad S_2 \qquad S_3}{\lceil\mathcal{D}\rceil_{\mathsf{type}}, \lceil\mathcal{D}\rceil_{\mathsf{constraint}}, \overline{\lceil\mathcal{C}\rceil}\sigma \Rightarrow \lceil\mathcal{D}\rceil_{\mathsf{clause}}}\,(\to\mathrm{L}, \wedge\mathrm{R})}{\overline{\lceil\mathcal{C}\rceil}\sigma \Rightarrow \overline{\lceil\mathcal{D}\rceil}}\,(\to\mathrm{R}, \wedge\mathrm{L})}{\lceil\mathcal{C}\rceil \Rightarrow \lceil\mathcal{D}\rceil}\,(\forall\mathrm{R}, \forall\mathrm{L})$$

where the sequents S_1, S_2 and S_3 are given by

$$\begin{aligned} S_1 &= \lceil\mathcal{D}\rceil_{\mathsf{type}} \Rightarrow \lceil\mathcal{C}\rceil_{\mathsf{type}}\sigma, \\ S_2 &= \lceil\mathcal{D}\rceil_{\mathsf{constraint}} \Rightarrow \lceil\mathcal{C}\rceil_{\mathsf{constraint}}\sigma, \\ S_3 &= \lceil\mathcal{C}\rceil_{\mathsf{clause}}\sigma \Rightarrow \lceil\mathcal{D}\rceil_{\mathsf{clause}}. \end{aligned}$$

It remains to show that the sequents S_1, S_2 and S_3 are provable in LKID-I. By Lemma 6.2 the sequent S_1 is provable in LKID-I. Consider the sequent S_2 and observe that every conjunct of $\lceil\mathcal{C}\rceil_{\mathsf{constraint}}\sigma$ occurs as a conjunct of $\lceil\mathcal{D}\rceil_{\mathsf{constraint}}$. Hence sequent S_2 is provable in LKID-I. Finally consider sequent S_3 and observe that it is of the form

$$l_1 \vee \cdots \vee l_n \vee (t = s)\sigma \vee (u = v)\sigma \Rightarrow l_1 \vee \cdots \vee l_n \vee (t = s)\sigma \vee (s = v)\sigma.$$

Hence the following is a proof of sequent S_3.

$$\dfrac{\dfrac{}{l_1 \Rightarrow l_1}\,(\mathrm{Ax}) \quad \ldots \quad \dfrac{}{l_n \Rightarrow l_n}\,(\mathrm{Ax}) \dfrac{\dfrac{\dfrac{}{\Rightarrow t\sigma = u\sigma}\,(=\mathrm{R})}{t\sigma = s\sigma, \Rightarrow s\sigma = u\sigma}\,(=\mathrm{L})}{t\sigma = s\sigma, u\sigma = v\sigma \Rightarrow s\sigma = v\sigma}\,(=\mathrm{L})}{S_3}\,(\vee\mathrm{L}, \vee\mathrm{R}, \mathrm{Wk})$$

This completes the proof of Lemma 6.12 □

6.2.2 Deductions

Having shown that the induction free system LKID-I simulates the three inference rules of the superposition calculus, we are now able to show that LKID-I simulates superposition deductions by applying Lemmas 6.10, 6.11, and 6.12 inductively.

Proof of Proposition 6.9. We proceed by order induction on the size of the of the superposition deduction of $\mathcal{C}$ from S and by case distinction on the last inference of the deduction.

If $\mathcal{C}$ is an initial clause, that is, $\mathcal{C} \in S$, then the formula $\lceil\mathcal{C}\rceil$ is a conjunct of $\lceil S\rceil$. Hence the sequent $\lceil S\rceil \Rightarrow \lceil\mathcal{C}\rceil$ is clearly provable in LKID-I.

If $\mathcal{C}$ is obtained via a superposition inference then there exist $i, j < n$ such that $\mathcal{C} = \sup(\mathcal{C}_i\alpha, \mathcal{C}_j\alpha)$ where α is a variable renaming such that $\mathcal{C}_i\alpha$ and $\mathcal{C}_j\alpha$ are variable disjoint. By the induction hypothesis the sequents $\mathrm{T}^{\mathsf{inj}}, \lceil S\rceil \Rightarrow \lceil\mathcal{C}_i\rceil$ and $\mathrm{T}^{\mathsf{inj}}, \lceil S\rceil \Rightarrow \lceil\mathcal{C}_j\rceil$ are provable in LKID-I. Thus the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S\rceil \Rightarrow \lceil\mathcal{C}_i, \mathcal{C}_j\rceil$ is also provable in LKID-I. Since α is a variable renaming and by the construction of $\lceil\cdot\rceil$ the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil\mathcal{C}_i, \mathcal{C}_j\rceil \Rightarrow \lceil\mathcal{C}_i\alpha, \mathcal{C}_j\alpha\rceil$ is easily seen to be provable in LKID-I. By Lemma 6.10 the sequent $\lceil\mathcal{C}_i\alpha, \mathcal{C}_j\alpha\rceil \Rightarrow \lceil\mathcal{C}_n\rceil$ is provable in LKID-I. Hence we obtain an LKID-I proof of $\mathrm{T}^{\mathsf{inj}}, \lceil S\rceil \Rightarrow \mathcal{C}$ as follows.

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \mathrm{T}^{\mathsf{inj}}, \lceil S\rceil \Rightarrow \lceil\mathcal{C}_i, \mathcal{C}_j\rceil\end{array} \qquad \begin{array}{c}\vdots\\ \lceil\mathcal{C}_i, \mathcal{C}_j\rceil \Rightarrow \lceil\mathcal{C}_i\alpha, \mathcal{C}_j\alpha\rceil\end{array}}{\mathrm{T}^{\mathsf{inj}}, \lceil S\rceil \Rightarrow \lceil\mathcal{C}_i\alpha, \mathcal{C}_j\alpha\rceil}\,(\mathrm{Cut}) \qquad \begin{array}{c}\vdots\\ \lceil\mathcal{C}_i\alpha, \mathcal{C}_j\alpha\rceil \Rightarrow \lceil\mathcal{C}\rceil\end{array}}{\mathrm{T}^{\mathsf{inj}}, \lceil S\rceil \Rightarrow \lceil\mathcal{C}\rceil}\,(\mathrm{Cut})$$

If $\mathcal{C}$ is obtained via a factorization inference then there exists $i < n$ such that $\mathcal{C} = \mathrm{fact}(\mathcal{C}_i)$. By the induction hypothesis the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S\rceil \Rightarrow \lceil\mathcal{C}_i\rceil$ is provable in LKID-I. By Lemma 6.12 the sequent $\lceil\mathcal{C}_i\rceil \Rightarrow \lceil\mathcal{C}\rceil$ is also provable in LKID-I. We thus obtain a proof of the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S\rceil \Rightarrow \lceil\mathcal{C}\rceil$ by applying a cut-inference to the two previous proofs.

If $\mathcal{C}$ is obtained via a reflection inference, then there exists $i < n$ such that $\mathcal{C} = \mathrm{refl}(\mathcal{C}_i)$. By the induction hypothesis the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S \rceil \Rightarrow \lceil \mathcal{C}_i \rceil$ is provable in LKID-I and by Lemma 6.11 the sequent $\lceil \mathcal{C}_i \rceil \Rightarrow \mathcal{C}$ is also provable in LKID-I. So by applying a cut-inference to these two proofs we immediately obtain a proof of $\mathrm{T}^{\mathsf{inj}}, \lceil S \rceil \Rightarrow \lceil \mathcal{C} \rceil$.

If $\mathcal{C}$ is obtained by normalization, then we apply Proposition 6.8 and we are done. □

The above Proposition can be formulated for the case where we consider an inference relation δ in the sense of Definition 3.24.

Proposition 6.13. *Let S be an L-clause set, δ an inference relation for S and $S_1, S_2 \subseteq S$. If $S_1 \vdash_\delta S_2$, then the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S[\top] \rceil, \lceil S_1 \rceil \Rightarrow \lceil S_2 \rceil$ is* LKID-I*-provable.*

Proof Sketch. Proceed by induction on the structure of the set $\{\mathcal{C} \in S \mid S_1 \vdash_\delta \mathcal{C}\}$. There are two base cases. If $S_1 \vdash_\delta \mathcal{C}$ because $\mathcal{C} \in S_1$, then by the definition of the translation $\lceil \cdot \rceil$ there is a conjunct $\lceil \mathcal{C} \rceil$ in the conjunction $\lceil S_1 \rceil$. Hence $\lceil S_1 \rceil \Rightarrow \lceil C \rceil$ is obviously provable in LKID-I. For the second base case suppose that $\mathcal{S}_1 \vdash_\delta \mathcal{C}$ because $\mathcal{C} \in S[\top]$. The conjunction $\lceil S[\top] \rceil$ clearly contains a conjunct of the form $\lceil \mathcal{C} \rceil$. Hence $\lceil S[\top] \rceil \Rightarrow \lceil \mathcal{C} \rceil$ is clearly provable in LKID-I. For the induction case suppose that $S_1 \vdash \mathcal{C}$ because $\delta(\mathcal{C})$ is defined and $S_1 \vdash \delta(\mathcal{C})$. By the induction hypothesis we obtain LKID-I proofs of the sequents $\mathrm{T}^{\mathsf{inj}}, \lceil S[\top] \rceil, \lceil S_1 \rceil \Rightarrow \lceil \mathcal{C}' \rceil$ for all $\mathcal{C}' \in \delta(\mathcal{C})$. Hence the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S[\top] \rceil, \lceil S_1 \rceil \Rightarrow \lceil \delta(\mathcal{C}) \rceil$ is also provable in LKID-I. Since the inference relation δ describes derivation in exactly one step we proceed by case distinction on the inference which derives $\mathcal{C}$ from clauses $\delta(\mathcal{C})$. By Lemmas 6.10, 6.11 and 6.12 we obtain in each case an LKID-I proof of the sequent $\lceil \delta(\mathcal{C}) \rceil \Rightarrow \lceil \mathcal{C} \rceil$. Hence the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S[\top] \rceil, \lceil S \rceil \Rightarrow \lceil \mathcal{C} \rceil$ is also provable in LKID-I. □

6.3 Inductive Cycles

In this section we will translate the inductive cycles of the n-clause calculus to proofs of the cyclic system CLKID^ω. We will start by discussing how an inductive cycle should be expressed on the sequent level and how it should be expressed in terms of cycles in the sequent calculus CLKID^ω. Having fixed these representations we carry out the actual translation.

6.3.1 Sequent Representation

As we have already mentioned in Section 3.2.2 the notion of inductive cycle of the n-clause calculus does not behave as an ordinary inference rule. It is better to view the detection of inductive cycles as a side condition for the termination of the refutation process. Thus we do not have an immediately obvious representation of inductive cycles in terms of sequents as we had for the inference rules of the superposition calculus. Nevertheless we can find a sequent representation for the information derived from the presence of inductive cycles. Remember that for a given clause set S the existence of an inductive cycle $(i, j, S_{\mathsf{init}}, S_{\mathsf{loop}})$ implies $S \models \eta \prec i$. Moreover it is this information

that is used to prove the "correctness" of the n-clause refutations. We thus only need to represent the information $\eta \prec i$ obtained from inductive cycles by sequents of the form

$$\mathsf{T}_\omega \eta, \lceil S \rceil \Rightarrow \lceil \eta \prec i \rceil.$$

6.3.2 Cyclic Proof Representation

Having decided which information is to be encoded in the sequents associated to inductive cycles, we need to decide how we would expect a proof of the correctness of inductive cycles to look like. That is, we need to decide which fragment of CLKID^ω we need to simulate inductive cycles. In Section 3.2.2 we have shown the correctness of inductive loops on the meta-level by a straightforward proof by infinite descent having an offset i and a constant descent step j. We have considered the j different cases; The cases 0 to $j-1$ are basically base cases that are proven without further descent and the j-th case was proven by constructing a smaller model. This proof pattern can be formalized as follows.

Definition 6.1. *A simple cyclic proof of a sequent $\Pi \Rightarrow \Lambda$ is a CLKID^ω proof π of the form:*

$$\dfrac{\dfrac{\dfrac{\vdots \pi^0_{\mathsf{base}}}{(\Gamma \Rightarrow \Delta)(0)}}{\Gamma, x_0 = 0 \Rightarrow \Delta}(=\mathrm{L}) \quad \ldots \quad \dfrac{\dfrac{\vdots \pi^{j-1}_{\mathsf{base}}}{(\Gamma \Rightarrow \Delta)(\overline{j-1})}}{\Gamma, x_0 = \overline{j-1} \Rightarrow \Delta}(=\mathrm{L}) \quad \dfrac{\dfrac{\dfrac{(\circlearrowleft)\ \Gamma, \underline{\mathsf{T}_\omega x_0} \Rightarrow \Delta}{\Gamma(x_j), \underline{\mathsf{T}_\omega x_j} \Rightarrow \Delta(x_j)}(\mathrm{Subst})}{\Gamma(\mathsf{s}^j x_j), \underline{\mathsf{T}_\omega x_j} \Rightarrow \Delta(\mathsf{s}^j x_j)}(=\mathrm{L})}{\Gamma, x_0 = \mathsf{s}^j x_j, \underline{\mathsf{T}_\omega x_j} \Rightarrow \Delta}(=\mathrm{L})}{(\circlearrowleft)\ \Gamma, \underline{\mathsf{T}_\omega x_0} \Rightarrow \Delta}(\mathrm{case}\mathsf{T}_\omega)^j$$

$$\dfrac{\vdots\ \pi_{\mathsf{end}}}{\Pi \Rightarrow \Lambda}$$

where the sequent $\Gamma, \mathsf{T}_\omega x_0 \Rightarrow \Delta$ contains only the free variable x_0; π^k_{base} with $0 \leq k < j$ are LKID-I *proofs; π_{step}, and π_{end} are* LKID-I *derivations. We say that $\Pi \Rightarrow \Lambda$ admits a simple cyclic proof with loop sequent $\Gamma, \mathsf{T}_\omega x_0 \Rightarrow \Delta$.*

Lemma 6.14. *Let S an L-clause set, δ an inference relation, $\sqsupseteq$ an immediate entailment relation and $(i, j, S_{\mathsf{init}}, S_{\mathsf{loop}})$ an inductive cycle for S with respect to δ and $\sqsupseteq$. Then there is a simple cyclic proof of the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S \rceil \Rightarrow \lceil \eta \prec i \rceil$.*

In order to prove this lemma we need to prove the following two lemmas about the $\downarrow$ operation.

Lemma 6.15. *Let $\mathcal{D}$ be an L-constraint clause, $j > 0$. The sequent*

$$\lceil \mathcal{D}\!\downarrow_j \rceil(\mathsf{s}^j \eta) \Rightarrow \lceil \mathcal{D} \rceil$$

is provable in LKID-I.

Proof. First of all note that the operation $\downarrow_j$ as well as the substitution of the variable η for $\mathsf{s}^j\eta$ affect only the constraints atoms and do not modify variables, clause literals, or type literals. Therefore we have

$$\lceil\mathcal{D}\rceil_{\mathsf{clause}} = \lceil\mathcal{D}{\downarrow_j}\rceil_{\mathsf{clause}}(\mathsf{s}^j\eta), \text{ and } \lceil\mathcal{D}\rceil_{\mathsf{sorts}} = \lceil\mathcal{D}{\downarrow_j}\rceil_{\mathsf{sorts}}(\mathsf{s}^j\eta).$$

Moreover since n-clauses are normalizable we may assume that $\mathcal{D}$ is of the form $[C \mid \eta \simeq t]$. Hence we can prove the sequent $\lceil\mathcal{D}{\downarrow_j}\rceil(\mathsf{s}^j\eta) \Rightarrow \lceil\mathcal{D}\rceil$ as follows:

$$\dfrac{\dfrac{}{\lceil C\rceil \Rightarrow \lceil C\rceil}\,(\mathrm{Ax}) \quad \dfrac{}{\lceil\mathcal{D}\rceil_{\mathsf{type}} \Rightarrow \lceil\mathcal{D}\rceil_{\mathsf{type}}}\,(\mathrm{Ax}) \quad \dfrac{\dfrac{}{\Rightarrow \mathsf{s}^j\eta = \mathsf{s}^j\eta}\,(=\mathrm{R})}{\eta = t \Rightarrow \mathsf{s}^j\eta = \mathsf{s}^j t}\,(=\mathrm{L})}{\overline{\lceil\mathcal{D}{\downarrow_j}\rceil}(\mathsf{s}^j\eta) \Rightarrow \overline{\lceil\mathcal{D}\rceil}}\,(\to\mathrm{L}, \to\mathrm{R}, \wedge\mathrm{R}, \wedge\mathrm{L})}{\lceil\mathcal{D}{\downarrow_j}\rceil(\mathsf{s}^j\eta) \Rightarrow \lceil\mathcal{D}\rceil}\,(\forall\mathrm{R}, \forall\mathrm{L})$$

□

Lemma 6.16. *Let S be an L-clause set, $j > 0$. The sequent*

$$\lceil S{\downarrow_j}\rceil(\mathsf{s}^j\eta) \Rightarrow \lceil S\rceil$$

is provable in LKID-I.

Proof. By repeated application of Lemma 6.15. □

We can now prove Lemma 6.14 as follows: Take the superposition deductions of the base cases and the step case of the cycle; obtain the respective translations; decompose the sequent $\lceil S\rceil \Rightarrow \lceil \eta \prec i\rceil$ and perform a j-fold case distinction; finally assemble all the proofs to obtain a simple cyclic proof.

Proof of Lemma 6.14. Since $(i, j, S_{\mathsf{init}}, S_{\mathsf{loop}})$ is an inductive cycle for S w.r.t. to δ and $\sqsupseteq$ we have:

$$S_{\mathsf{init}} \vdash_\delta \eta \not\approx k, \text{ for all } k \text{ with } i \leq k \leq i + j,$$
$$S_{\mathsf{init}} \vdash_\delta S_{\mathsf{loop}},$$
$$S_{\mathsf{loop}} \vdash S_{\mathsf{init}}{\downarrow_j}\ .$$

By Lemmas 6.9 and 6.13 there exist LKID-I proofs $\pi^0_{\mathsf{base}}, \ldots, \pi^{i+j-1}_{\mathsf{base}}, \pi^0_{\mathsf{step}}, \pi^1_{\mathsf{step}}$ such that:

$$\pi^k_{\mathsf{base}} : \mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\rceil \Rightarrow \lceil \eta \not\approx k\rceil, \text{ for all } k \text{ with } i \leq k < i + j,$$
$$\pi^0_{\mathsf{step}} : \mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\rceil \Rightarrow \lceil S_{\mathsf{loop}}\rceil,$$
$$\pi^1_{\mathsf{step}} : \mathrm{T}^{\mathsf{inj}}, \lceil S_{\mathsf{loop}}\rceil \Rightarrow \lceil S_{\mathsf{init}}{\downarrow_j}\rceil.$$

By Lemma 6.16 there exists an LKID-I proof γ of the sequent

$$\lceil S[\top]\rceil, \lceil S_{\mathsf{init}}{\downarrow_j}\rceil(\mathsf{s}^{i+j}\nu_j) \Rightarrow \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i\nu_j).$$

Define the CLKID$^\omega$ derivation π recursively. The initial piece of the derivation π is as follows:

$$\dfrac{\dfrac{}{\Rightarrow \nu_0 = \nu_0}\,(=\mathrm{R}) \qquad \begin{array}{c}\vdots\,\varphi_0\\ \mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i\nu_0), \underline{\mathsf{T}_\omega\nu_0}, \nu_0 = \nu_0 \Rightarrow\end{array}}{\dfrac{\dfrac{(\circlearrowleft) \quad \mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i\nu_0), \underline{\mathsf{T}_\omega\nu_0} \Rightarrow}{\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\rceil, \mathsf{T}_\omega\nu_0, \eta = \mathsf{s}^i\nu_0 \Rightarrow}\,(=\mathrm{L})}{\mathrm{T}^{\mathsf{inj}}, \lceil S\rceil \Rightarrow \lceil \eta \prec i\rceil.}\,(\forall\mathrm{R}, \to\mathrm{R}, \wedge\mathrm{L}, \mathrm{Wk})}\,(\mathrm{Cut})$$

The LKID-I proofs ψ_k with $0 \leq k < j$ represent the base cases of the cyclic proof π. These proofs are as shown below.

$$\dfrac{\dfrac{\begin{array}{c}\vdots\,\pi^k_{\mathsf{base}}\\ \mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\rceil \Rightarrow \eta \neq \overline{k}\end{array}}{\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\rceil(\overline{k}) \Rightarrow \overline{k} \neq \overline{k}}\,(\mathrm{Subst}) \qquad \dfrac{\dfrac{}{\Rightarrow \overline{k} = \overline{k}}\,(=\mathrm{R})}{\overline{k} \neq \overline{k} \Rightarrow}\,(\neg\mathrm{L})}{\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\rceil(\overline{k}) \Rightarrow}\,(\mathrm{Cut})$$

The derivations φ_k with $0 \leq k < j$ represent the successive case distinctions of our argument by infinite descent. For brevity we set $\Gamma = \{\mathrm{T}^{\mathsf{inj}}\} \cup \lceil S[\top]\rceil \cup \lceil S_{\mathsf{init}}\rceil$. These proofs are as follows:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\,\psi_k\\ \Gamma(\overline{k}) \Rightarrow\end{array}}{\Gamma, \nu_0 = \overline{k} \Rightarrow}\,(=\mathrm{L})}{\Gamma(\mathsf{s}^i\nu_0), \nu_k = \mathsf{0}, \nu_0 = \mathsf{s}^k\nu_k \Rightarrow}\,(=\mathrm{L}) \qquad \dfrac{\begin{array}{c}\vdots\,\varphi_{k+1}\\ \Gamma(\mathsf{s}^i\nu_0), \underline{\mathsf{T}_\omega\nu_{k+1}}, \nu_0 = \mathsf{s}^{k+1}\nu_{k+1} \Rightarrow\end{array}}{\Gamma(\mathsf{s}^i\nu_0), \underline{\mathsf{T}_\omega\nu_{k+1}}, \nu_k = s\nu_{k+1}, \nu_0 = \mathsf{s}^k\nu_k \Rightarrow}\,(=\mathrm{L})}{\Gamma(\mathsf{s}^i\nu_0), \underline{\mathsf{T}_\omega\nu_k}, \nu_0 = \mathsf{s}^k\nu_k \Rightarrow}\,(\mathrm{Case}\mathsf{T}_\omega)$$

The derivation φ_j of the descent step makes use of the proofs ε_1 and ε_2 given below, respectively. The proofs ε_1 and ε_2 introduce the formulas $\lceil S_{\mathsf{loop}}\rceil(\mathsf{s}^{i+j}\nu_j)$ and $\lceil S_{\mathsf{init}}\downarrow_j\rceil(\mathsf{s}^{i+j}\nu_j)$, respectively. The latter formula is then used to reach the loop sequent.

$$\dfrac{\begin{array}{c}\vdots\,\pi^0_{\mathsf{step}}\\ \mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\rceil \Rightarrow \lceil S_{\mathsf{loop}}\rceil\end{array}}{\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\rceil(\mathsf{s}^{i+j}\nu_j) \Rightarrow \lceil S_{\mathsf{loop}}\rceil(\mathsf{s}^{i+j}\nu_j)}\,(\mathrm{Subst})$$

$$\dfrac{\begin{array}{c}\vdots\,\pi^1_{\mathsf{step}}\\ \mathrm{T}^{\mathsf{inj}}, \lceil S_{\mathsf{loop}}\rceil \Rightarrow \lceil S_{\mathsf{init}}\downarrow_j\rceil\end{array}}{\mathrm{T}^{\mathsf{inj}}, \lceil S_{\mathsf{loop}}\rceil(\mathsf{s}^{i+j}\nu_j) \Rightarrow \lceil S_{\mathsf{init}}\downarrow_j\rceil(\mathsf{s}^{i+j}\nu_j)}\,(\mathrm{Subst})$$

The derivation φ_j is then formed as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{(\circlearrowleft)\quad \mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i\nu_0), \underline{\mathsf{T}_\omega\nu_0} \Rightarrow}{\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i\nu_j), \underline{\mathsf{T}_\omega\nu_j} \Rightarrow}\,(\mathrm{Subst}) \\ \vdots\gamma \\ \varepsilon_2 \quad \mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\downarrow_j\rceil(\mathsf{s}^{i+j}\nu_j), \underline{\mathsf{T}_\omega\nu_j} \Rightarrow}{\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{loop}}\rceil(\mathsf{s}^{i+j}\nu_j), \underline{\mathsf{T}_\omega\nu_j} \Rightarrow}\,(\mathrm{Cut}) \\ \varepsilon_1 \quad \mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\rceil(\mathsf{s}^{i+j}\nu_j), \underline{\mathsf{T}_\omega\nu_j} \Rightarrow}{\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \lceil S_{\mathsf{init}}\rceil(\mathsf{s}^{i}\nu_0), \underline{\mathsf{T}_\omega\nu_j}, \nu_0 = \mathsf{s}^j\nu_j \Rightarrow}\,(=\mathrm{L})$$

It remains to show that the derivation π is a simple cyclic proof of the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S\rceil \Rightarrow \lceil \eta \prec i\rceil$. The derivation π clearly is a CLKID^ω pre-proof. Moreover π contains exactly one infinite path and this path clearly has infinitely many progress points. Thus, π satisfies the trace condition and therefore it is a CLKID^ω proof. By its construction the proof π is easily seen to be a simple cyclic proof. □

6.4 Simple Cyclic Proofs

In this section we will show that the notion of simple cyclic proof is simulated by the related notion of *simple induction proof*. Indeed both notions are equivalent but showing simulation in the one direction is sufficient for our purposes. Simple induction proofs are induction proofs that contain only a single induction inference. Instead of showing only the particular case that the cyclic proof representation of an inductive cycle is provable in LKID-I we will show slightly a more general case in the hope that this more general result might come in handy when we want to consider a different cyclic proof representation of inductive cycles.

Definition 6.2. *Let $j \in \mathbb{N}$ with $j > 0$. A j-step simple induction proof of a sequent $\Pi \Rightarrow \Lambda$ is an* LKID *proof π of the form:*

$$\dfrac{\begin{array}{cccc}\vdots\pi^0_{\mathsf{base}} & \vdots\pi^{j-1}_{\mathsf{base}} & \vdots\pi_{\mathsf{step}} & \vdots\pi_{\mathsf{cut}} \\ \Gamma \Rightarrow H(\mathsf{0}), \Delta \quad \ldots \quad \Gamma \Rightarrow H(\overline{j-1}), \Delta & \Gamma, H(\nu) \Rightarrow H(\mathsf{s}^j\nu), \Delta & \Gamma, H(x_0) \Rightarrow \Delta\end{array}}{\Gamma, \mathsf{T}_\omega x_0 \Rightarrow \Delta}\,(\mathrm{Ind}\mathsf{T}_\omega)$$
$$\vdots\pi_{\mathsf{end}}$$
$$\Pi \Rightarrow \Lambda$$

where the sequent $\Gamma, \mathsf{T}_\omega x_0 \Rightarrow \Delta$ has only the free variable x_0; π^k_{base} with $0 \leq k < j-1$, π_{step} and π_{cut} are LKID-I *proofs; π_{end} is an* LKID-I *derivation.*

We are interested in translating simple cyclic proofs to simple induction proofs as defined above. The major part of this task is the invention of a suitable induction hypothesis, that allows us to prove the base cases, step case, and the major premise. Let

us start by discussing informally what the induction hypothesis will look like. If we look at the shape of a simple cyclic proof we can see that it is already syntactically very close to a simple induction proof. The proofs π^k_{base} with $k = 1, \ldots, j-1$ and their respective end-sequents

$$(\Gamma \Rightarrow \Delta)(\overline{k})$$

remind of induction base cases and their respective proofs. Moreover the derivation

$$\begin{array}{c} \Gamma(x_j), \mathsf{T}_\omega x_j \Rightarrow \Delta(x_j) \\ \vdots \pi_{\mathsf{step}} \\ \Gamma(\mathsf{s}^j x_j), \mathsf{T}_\omega(x_j) \Rightarrow \Delta(\mathsf{s}^j x_j) \end{array}$$

reminds of the proof of an induction step. The idea is thus to use the formula corresponding to the loop sequent as the induction hypothesis for the simple cyclic proof. However, there is one detail that we need to take care of. The explicit occurrence of the formula $\mathsf{T}_\omega x_j$ in the end-sequent of the derivation π^{step} does not fit into this scheme. If we would take the formula representation of the loop sequent as the induction hypothesis, then the variable x_j would have to appear in the context of a term $\mathsf{s}^j x_j$. However we can observe that the formula occurrence $\mathsf{T}_\omega x_j$ occupies a special place in the simple cyclic proof – it is the traced formula. Its purpose is thus to keep track of the types and the progress points, besides this it may of course also occur in other parts of the derivation.

The arguments by infinite descent represented by cycles in CLKID^ω are thus always accompanied by type information for the traced variable. This is in contrast to the induction rules which do not dispose of this type information. This seems a little bit curious at first, but it is not hard to see that we can obtain this type information in inductive proofs by carrying out a second induction, that runs in parallel to the main induction, and whose purpose is to inherit type information. Such a "parallel" induction is obtained by adding the conjunct $\mathsf{T}_\omega x_0$ to the induction hypothesis of the main induction.

Definition 6.3. *Let π be a simple cyclic proof with loop sequent $\Pi, \mathsf{T}_\omega x_0 \Rightarrow \Lambda$. The formula $H_\pi(x_0)$ is given by*

$$H_\pi(x_0) = \mathsf{T}_\omega x_0 \wedge (\bigwedge \Pi' \rightarrow \bigvee \Lambda'),$$

where Π' and Λ' are all the formulas of Π and Λ respectively, containing a free occurrence of the variable x_0.

Lemma 6.17. *Let $\Gamma \Rightarrow \Delta$ be a sequent. If S admits a j-step simple cyclic proof, then S admits a j-step simple induction proof with induction invariant $H_\pi(x_0)$.*

We delay the proof of Lemma 6.17 until we have shown that we can obtain all of the required components for the desired simple induction proof.

Lemma 6.18. *Let π be a j-step simple cyclic proof with loop sequent $\Pi(x_0), \mathsf{T}_\omega x_0 \Rightarrow \Lambda(x_0)$ and let $k \in \{0, \dots, j-1\}$. Then the sequent $\Pi \Rightarrow H_\pi(\overline{k}), \Lambda$ is provable in* LKID-I.

Proof. By Definition 6.3 the formula $H_\pi(x_0)$ is necessarily of the form:

$$\mathsf{T}_\omega \overline{k} \wedge (\bigwedge \Pi_2 \to \bigvee \Lambda_2),$$

where Π_2 and Λ_2 are all the formulas of Π and Λ respectively that contain a free occurrence of the variable x_0. We set $\Pi_1 = \Pi \setminus \Pi_2$ and analogously we define Λ_1. By Lemma 6.1 the sequent $\Rightarrow \mathsf{T}_\omega \overline{k}$ is provable in LKID-I. Hence the following is a proof of the sequent $\Pi \Rightarrow H_\pi(\overline{k}), \Lambda$.

$$\dfrac{\Rightarrow \mathsf{T}_\omega \overline{k} \qquad \dfrac{\vdots \pi^k_{\mathsf{base}}}{\dfrac{\dfrac{\Pi_1, \Pi_2(\overline{k}) \Rightarrow \Lambda_1, \Lambda_2(\overline{k})}{\Pi_1 \Rightarrow \bigwedge \Pi_2(\overline{k}) \to \bigvee \Lambda_2(\overline{k}), \Lambda_1}\,(\to\mathrm{R}, \wedge\mathrm{L}, \vee\mathrm{R})}{\Pi \Rightarrow \bigwedge \Pi_2(\overline{k}) \to \bigvee \Lambda_2(\overline{k}), \Lambda}\,(\mathrm{Wk})}}{\Pi \Rightarrow \mathsf{T}_\omega \overline{k} \wedge (\bigwedge \Pi_2(\overline{k}) \to \bigvee \Lambda_2(\overline{k})), \Lambda}\,(\wedge\mathrm{R})$$

□

Lemma 6.19. *Let π be a j-step simple cyclic proof with loop sequent $\Pi(x_0), \mathsf{T}_\omega x_0 \Rightarrow \Lambda(x_0)$. Then the sequent $\Pi, H_\pi(x_0) \Rightarrow \Lambda$ is provable in* LKID-I.

Proof. We define the sets of formulas Π_1, Π_2, Λ_1, and Λ_2 as in the proof of Lemma 6.4. Then the following is a proof of the sequent $\Pi, H_\pi(x_0) \Rightarrow \Lambda$.

$$\dfrac{\dfrac{\dfrac{\vdots \qquad \vdots}{\Pi_2 \Rightarrow \bigwedge \Pi_2 \qquad \bigvee \Lambda_2 \Rightarrow \Lambda}\,(\to\mathrm{L})}{\Pi, \bigwedge \Pi_2 \to \bigvee \Lambda_2 \Rightarrow \Lambda}\,(\wedge\mathrm{L}, \mathrm{Wk})}{\Pi, \mathsf{T}_\omega x_0 \wedge (\bigwedge \Pi_2 \to \bigvee \Lambda_2) \Rightarrow \Lambda}$$

□

Finally, we need to show that the induction step is provable for the induction invariant $H_\pi(x_0)$.

Lemma 6.20. *Let π be a j-step simple cyclic proof with loop sequent $\Pi(x_0), \mathsf{T}_\omega x_0 \Rightarrow \Lambda(x_0)$. Then the sequent $\Pi, H_\pi(\nu) \Rightarrow H_\pi(\mathsf{s}^j \nu), \Lambda$ is provable in* LKID-I.

We prove this lemma by constructing a suitable proof from the derivation π_{step}. The idea is to pass the induction hypothesis in the contexts up to the bud and to use it to "close" the bud. Care must be taken that adding the formula to the contexts of the sequents on the path from the to the bud does not conflict with any strong quantifier inferences.

Proof of Lemma 6.20. Define Π_1, Π_2, Λ_1, and Λ_2 as in the proof of Lemma 6.4. By Lemma 6.19 there exists an LKID-I proof γ_{cut} of the sequent $\Pi, H_\pi(x_0) \Rightarrow \Lambda$. By lemma 6.1 the sequent $\mathsf{T}_\omega \nu \Rightarrow \mathsf{T}_\omega s^j \nu$ is provable in LKID-I.

Consider the derivation π'_{step} obtained by adding the formula $H_\pi(x_j)$ to the antecedent of the sequents in π_{step} that belong to the path to the bud. We need to show that π'_{step} is indeed a derivation of LKID-I. It suffices to show that this operation does not interfere with strong quantifier inferences. By its construction, the formula $H_\pi(x_j)$ has only the free variable x_j. Observe that the explicit occurrence of the formula $\mathsf{T}x_j$ in the end-sequent of π_{step} is the traced formula. Since the proof π is a CLKID^ω proof, it satisfies the trace condition. Hence every sequent on the path to the bud of π_{step} contains an ancestor of the formula occurrence of $\mathsf{T}x_j$. Therefore the variable x_j occurs freely in all the sequents on this path. Hence there can be no strong quantifier inferences on this path. Thus the derivation π'_{step} is an LKID derivation of the form

$$\begin{array}{c} H_\pi(x_j), \Pi(x_j), \mathsf{T}_\omega x_j \Rightarrow \Lambda(x_j) \\ \vdots \\ H_\pi(x_j), \Pi(\mathsf{s}^j x_j), \mathsf{T}_\omega x_j \Rightarrow \Lambda(\mathsf{s}^j x_j). \end{array}$$

We can then prove the sequent $\Pi, H_\pi(\nu) \Rightarrow H_\pi(\mathsf{s}^j \nu), \Lambda$ as follows

$$\dfrac{\begin{array}{c} \mathsf{T}_\omega \nu \Rightarrow \mathsf{T}_\omega \mathsf{s}^j \nu \end{array} \quad \dfrac{\dfrac{\dfrac{\vdots \gamma_{\mathsf{cut}}}{H_\pi(x_j), \Pi(x_j) \Rightarrow \Lambda(x_j)}\,(\text{Subst})}{H_\pi(x_j), \mathsf{T}_\omega x_j, \Pi(x_j) \Rightarrow \Lambda(x_j)} \\ \vdots \pi'_{\mathsf{step}} \\ \dfrac{\dfrac{H_\pi(x_j), \mathsf{T}_\omega x_j, \Pi(\mathsf{s}^j x_j) \Rightarrow \Lambda(\mathsf{s}^j x_j)}{H_\pi(\nu), \mathsf{T}_\omega \nu, \Pi(\mathsf{s}^j \nu) \Rightarrow \Lambda(\mathsf{s}^j \nu)}\,(\text{Subst})}{\Pi, \mathsf{T}_\omega \nu, H_\pi(\nu) \Rightarrow \bigwedge \Pi_2(\mathsf{s}^j \nu) \to \bigvee \Lambda_2(\mathsf{s}^j \nu), \Lambda}\,(\to\text{R}, \wedge\text{L}, \vee\text{R})}{\Pi, \mathsf{T}_\omega \nu \wedge (\bigwedge \Pi_2(\nu) \to \bigvee \Lambda_2(\nu)) \Rightarrow \mathsf{T}_\omega \mathsf{s}^j \nu \wedge (\bigwedge \Pi_2(\mathsf{s}^j \nu) \to \bigvee \Lambda_2(\mathsf{s}^j \nu)), \Lambda}\,(\wedge\text{L}, \wedge\text{R}, \text{Wk})$$

□

Proof of Lemma 6.17. Let π be a j-step simple cyclic proof of the sequent $\Gamma \Rightarrow \Delta$ with loop sequent $\Pi, \mathsf{T}_\omega x_0 \Rightarrow \Lambda$. By Lemma 6.4 there exists an LKID-I proof γ^k_{base} with end-sequent $\Pi \Rightarrow H_\pi(\overline{k}), \Lambda$ for $k = 0, \ldots, j-1$. By Lemma 6.19 there exists an LKID-I proof γ_{cut} with end-sequent $\Pi, H_\pi(x_0) \Rightarrow \Lambda$ and by Lemma 6.20 there exists an LKID-I proof γ_{step} of the sequent $\Pi, H_\pi(\nu) \Rightarrow H_\pi(\mathsf{s}^j \nu), \Lambda$. Hence the following is a simple induction proof of the sequent $\Gamma \Rightarrow \Delta$.

$$\dfrac{\begin{array}{cccc}\vdots\gamma^1_{\mathsf{base}} & & \vdots\gamma^k_{\mathsf{base}} & \vdots\gamma_{\mathsf{step}} \qquad \vdots\gamma_{\mathsf{cut}} \\ \Pi \Rightarrow H_\pi(\overline{1}), \Lambda & \ldots & \Pi \Rightarrow H(\overline{k}), \Lambda & \Pi, H_\pi(\nu) \Rightarrow H_\pi(\mathsf{s}^j\nu), \Lambda \quad \Pi, H_\pi(x_0) \Rightarrow \Lambda\end{array}}{\begin{array}{c}\Pi, \mathsf{T}_\omega x_0 \Rightarrow \Lambda \\ \vdots\pi_{\mathsf{end}} \\ \Gamma \Rightarrow \Delta\end{array}}\,(\mathrm{Ind}\mathsf{T}_\omega)$$

□

It might be interesting to observe that there is some form of redundancy in the simple cyclic proof constructed above. Indeed the proof of the major-premise also occurs as a subproof in the proof of the step case.

6.5 Refutations of the n-Clause Calculus

Previously, we have shown that every simple cyclic proof corresponds to a simple induction proof. Moreover, we have already shown that inductive cycles of the n-clause calculus can be represented as simple cyclic proofs. We can thus obtain a simple induction proof representing an inductive cycle. We will show that we can improve the induction hypothesis a bit. Finally we will put the previously obtained simple inductive proofs to use, in order to obtain inductive proofs that simulate n-clause refutations.

Corollary 6.21. *Let S be an L-clause set, δ an inference relation for S and $\sqsupseteq$ an immediate entailment relation for S. If S admits an inductive cycle $(i, j, S_{\mathsf{init}}, S_{\mathsf{loop}})$, then the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S \rceil \Rightarrow \lceil \eta \prec i \rceil$ admits a j-step simple induction proof with induction hypothesis $\mathsf{T}_\omega\eta \wedge \neg\lceil S_{\mathsf{init}} \rceil(\mathsf{s}^i\eta)$.*

Proof. By Lemma 6.14 the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S \rceil \Rightarrow \lceil \eta \prec i \rceil$ admits a j-step simple cyclic proof with loop sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S[\top] \rceil, \lceil S_{\mathsf{init}} \rceil(\mathsf{s}^i\nu_0), \mathsf{T}_\omega\nu_0 \Rightarrow$. Hence by Lemma 6.17 the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S \rceil \Rightarrow \lceil \eta \prec i \rceil$ admits a j-step simple induction proof with induction hypothesis $\mathsf{T}_\omega\eta \wedge \neg\lceil S_{\mathsf{init}} \rceil(\mathsf{s}^i\eta)$. □

We have observed that the type atom $\mathsf{T}_\omega\eta$ is required in order to provide the induction with the type information that is used by the cyclic proof. However the cyclic proof constructed in the proof of Lemma 6.14 does not use this type information. It should thus be possible to eliminate this conjunct from the induction invariant.

Corollary 6.22. *Let S be as in Corollary 6.21. If S admits an inductive cycle (i, j, S_{init}, S_{loop}), then the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S \rceil \Rightarrow \lceil \eta \prec i \rceil$ admits a simple induction proof with induction invariant $\neg\lceil S_{\mathsf{init}} \rceil(\mathsf{s}^i\eta)$.*

Proof. Denote by π the simple cyclic proof constructed by the proof of Lemma 6.14 and denote by γ the simple induction proof obtained by applying the construction of the proof of Lemma 6.17 to π. We will define a global transformation on γ that yields the

desired proof. Consider any proof γ^k_{base} with $0 \leq k < j$. The proof γ^k_{base} is without loss of generality of the form

$$\dfrac{\begin{array}{cc}\vdots & \vdots\pi^k_{\mathsf{step}} \\ \Rightarrow \mathsf{T}_\omega\overline{i+k} & \mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil\lceil S_{\mathsf{init}}\rceil(\overline{i+k}) \Rightarrow\end{array}}{\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil \Rightarrow \mathsf{T}_\omega\overline{i+k} \wedge \neg\lceil S_{\mathsf{init}}\rceil(\overline{i+k}).}\,(\wedge\mathrm{R}, \neg\mathrm{R}, \mathrm{Wk})$$

Hence there is an LKID-I proof of the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil \Rightarrow \neg\lceil S_{\mathsf{init}}\rceil(\overline{i+k})$.

Next let us consider the proof γ_{cut}. By its construction the formula $\mathsf{T}_\omega x_0$ only appears in the contexts of the inferences. Hence there is an LKID-I proof γ'_{cut} of the sequent $\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \neg\lceil S_{\mathsf{init}}\rceil(s^i x_0) \Rightarrow$.

Now consider the proof π_{step}. By the construction this proof is of the form:

$$\dfrac{\dfrac{\vdots\gamma_{\mathsf{cut}}}{\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \mathsf{T}_\omega x_0, \mathsf{T}_\omega x_0 \wedge \neg\lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i x_0), \lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i x_0) \Rightarrow}\,(\mathrm{Subst})}{\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \mathsf{T}_\omega\nu, \mathsf{T}_\omega\nu \wedge \neg\lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i\nu), \lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i\nu) \Rightarrow}$$

$$\dfrac{\begin{array}{cc}\vdots & \vdots\gamma' \\ \mathsf{T}_\omega\nu \Rightarrow \mathsf{T}_\omega\mathsf{s}^i\nu & \mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \mathsf{T}_\omega\nu, \mathsf{T}_\omega\nu \wedge \neg\lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i\nu), \lceil S_{\mathsf{init}}\rceil(\mathsf{s}^{i+j}\nu) \Rightarrow\end{array}}{\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \mathsf{T}_\omega\nu \wedge \neg\lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i\nu) \Rightarrow \mathsf{T}_\omega\mathsf{s}^j\nu \wedge \neg\lceil S_{\mathsf{init}}\rceil(\mathsf{s}^{i+j}\nu)}\,(\wedge\mathrm{R}, \neg\mathrm{R}, \mathrm{Wk})$$

By the same arguments as in the proof of Lemma 6.1 it is possible to replace the formula occurrences of $\mathsf{T}_\omega\nu \wedge \neg\lceil S_{\mathsf{init}}\rceil(\mathsf{s}\nu)$ by $\neg\lceil S_{\mathsf{init}}\rceil(s^i\nu)$ in the derivation γ' on the path to the bud. We denote the resulting derivation by γ''. Furthermore by the construction of the proof π_{step} the ancestors of the explicit formula occurrence $\mathsf{T}_\omega\nu$ in the end-sequent of π_{step} do only occur in contexts of γ''. Hence we can remove these formula occurrences from γ'' to in order to obtain a derivation γ'''. We can then prove the step sequent as follows:

$$\dfrac{\dfrac{\vdots\gamma'_{\mathsf{cut}}}{\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \neg\lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i x_0), \lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i x_0) \Rightarrow}\,(\mathrm{Subst})}{\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \neg\lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i\nu), \lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i\nu) \Rightarrow}$$

$$\dfrac{\dfrac{\vdots\gamma'''}{\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \neg\lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i\nu), \lceil S_{\mathsf{init}}\rceil(\mathsf{s}^{i+j}\nu) \Rightarrow}\,(\neg\mathrm{R})}{\mathrm{T}^{\mathsf{inj}}, \lceil S[\top]\rceil, \neg\lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i\nu) \Rightarrow \neg\lceil S_{\mathsf{init}}\rceil(\mathsf{s}^{i+j}\nu)}$$

□

Our aim is the representation of inductive cycles with respect to derivability in LKID-I under a set of axioms.

Corollary 6.23. *Let S be an L-clause set, δ an inference relation for S and $\sqsupseteq$ an immediate entailment relation for S. If S admits an inductive cycle $(i, j, S_{\mathsf{init}}, S_{\mathsf{loop}})$, then the sequent $\mathrm{I}^{j}_{\eta}\neg\lceil S_{\mathsf{init}}\rceil(\mathsf{s}^{i}\eta), \mathrm{T}^{\mathsf{inj}}, \lceil S\rceil \Rightarrow \lceil \eta \prec i\rceil$ is provable in* LKID-I.

Proof. Immediate. □

As stated in Definition 3.29 a set of n-clauses is considered as refuted if it contains a cycle with offset i (i.e. S entails $\eta \prec i$) and the clauses $\eta \not\approx 0, \ldots, \eta \not\approx i-1$ are derivable from S. We will thus show that the formula $\lceil S\rceil$ is unsatisfiable by showing that the sequent $\lceil \eta \not\approx 0\rceil, \ldots, \lceil \eta \not\approx i-1\rceil, \lceil \eta \prec i\rceil \Rightarrow$ is provable. To prove this sequent we need the information that every element of T_ω is either of the form $\mathsf{0}$, or $\overline{1}$, or $\ldots$, or $\overline{i-1}$ or the i-th successor of some other element.

Definition 6.4. *Let $i \in \mathbb{N}$ with $i > 0$, then the formulas $\mathsf{cd}_i(x)$, $\mathsf{case}_i(x)$, and Case_i are given by*

$$\mathsf{cd}_i(x) = \bigvee_{j=0}^{i-1} x = \overline{j} \vee (\mathsf{T}_\omega y \wedge \mathsf{s}^i y = x),$$
$$\mathsf{case}_i(x) = \exists y(\mathsf{T}_\omega x \to (\mathsf{cd}_i(x))),$$
$$\mathsf{Case}_i(x) = \forall x\ \mathsf{case}_i(x).$$

Lemma 6.24. *The formula $\mathsf{T}_\omega x \wedge \mathsf{case}_1(x)$ is provable in* LKID *with induction invariant $\mathsf{T}_\omega x \wedge \mathsf{case}_1(x)$.*

Proof. The base case is proved as follows

$$\dfrac{\dfrac{}{\Rightarrow \mathsf{T}_\omega 0}\,(\mathsf{T}_\omega \mathrm{R}_1) \qquad \dfrac{}{\Rightarrow \mathsf{0} = \mathsf{0}}\,(=\mathrm{R})}{\Rightarrow \mathsf{T}_\omega 0 \wedge (\mathsf{T}_\omega \mathsf{0} \to \exists y(\mathsf{0} = \mathsf{0} \vee (\mathsf{T}_\omega y \wedge \mathsf{s}y = \mathsf{0}))).}\,(\wedge\mathrm{R}, \to\mathrm{R}, \exists\mathrm{R}, \vee\mathrm{R})$$

The induction step can be proved as follows

$$\dfrac{\begin{array}{c}\vdots\varepsilon\\ \mathsf{T}_\omega x \Rightarrow \mathsf{T}_\omega \mathsf{s}x \qquad \mathsf{T}_\omega x, \mathsf{case}_1(x) \Rightarrow \mathsf{case}_1(\mathsf{s}x)\end{array}}{\mathsf{T}x \wedge \mathsf{case}_1(x) \Rightarrow \mathsf{T}_\omega \mathsf{s}x \wedge \mathsf{case}_1(\mathsf{s}x).}\,(\wedge\mathrm{R}, \wedge\mathrm{L}, \mathrm{Wk})$$

where the proof ε and its subproofs ε_1 and ε_2 are given below, respectively.

$$\dfrac{\dfrac{}{\mathsf{T}_\omega x \Rightarrow \mathsf{T}_\omega x}\,(\mathrm{Ax}) \qquad \dfrac{\begin{array}{c}\varepsilon_1 \qquad \varepsilon_2\\ \mathsf{T}x, \exists y(x = \mathsf{0} \vee (\mathsf{T}_\omega y \wedge \mathsf{s}y = x)) \Rightarrow \mathsf{case}_1(\mathsf{s}x)\end{array}}{}\,(\exists\mathrm{L}, \vee\mathrm{L}, \mathrm{Wk})}{\mathsf{T}x, \mathsf{T}x \to \exists y(x = \mathsf{0} \vee (\mathsf{T}_\omega y \wedge \mathsf{s}y = x)) \Rightarrow \mathsf{case}_1(\mathsf{s}x)}\,(\to\mathrm{L})$$

$$\dfrac{\dfrac{\dfrac{}{\Rightarrow \mathsf{T}_\omega \mathsf{0}}\,(\mathsf{T}\mathrm{R}_1) \qquad \dfrac{\dfrac{}{\Rightarrow \mathsf{s0} = \mathsf{s0}}\,(=\mathrm{R})}{x = \mathsf{0} \Rightarrow \mathsf{s0} = \mathsf{s}x}\,(=\mathrm{L})}{x = \mathsf{0} \Rightarrow \mathsf{T}_\omega \mathsf{0} \wedge \mathsf{s0} = \mathsf{s}x}\,(\wedge\mathrm{R})}{x = \mathsf{0} \Rightarrow \mathsf{case}_1(\mathsf{s}x)}\,(\to\mathrm{R}, \exists\mathrm{R}, \vee\mathrm{R}, \mathrm{Wk})$$

$$\dfrac{\dfrac{\dfrac{}{\mathsf{T}_\omega\beta \Rightarrow \mathsf{T}_\omega\beta}\,(\mathrm{Ax})}{\mathsf{T}_\omega\beta \Rightarrow \mathsf{T}_\omega\mathsf{s}\beta}\,(\mathrm{TR}_2) \qquad \dfrac{\dfrac{}{\Rightarrow \mathsf{ss}\beta = \mathsf{ss}\beta}\,(\mathrm{Ax})}{s\beta = x \Rightarrow \mathsf{ss}\beta = \mathsf{s}x}\,(=\mathrm{L})}{\mathsf{T}_\omega\beta \wedge \mathsf{s}\beta = x \Rightarrow \mathsf{T}_\omega\mathsf{s}\beta \wedge \mathsf{ss}\beta = \mathsf{s}x}\,(\wedge\mathrm{R}, \wedge\mathrm{L})}{\mathsf{T}_\omega\beta \wedge \mathsf{s}\beta = x \Rightarrow \mathsf{case}_1(\mathsf{s}x)}\,(\to\mathrm{R}, \exists\mathrm{R}, \vee\mathrm{R}, \mathrm{Wk})$$

□

Lemma 6.25. *Let i be a natural number with $i \geq 1$. The sequent $\mathsf{Case}_1 \Rightarrow \mathsf{Case}_i$ is provable in* LKID-I.

Proof. We proceed by weak induction on i. The case where $i = 1$ is trivial. If $i > 1$ then by the induction hypothesis there is a proof π of the sequent $\mathsf{Case}_1 \Rightarrow \mathsf{Case}_{i-1}$. We denote by γ the proof below and ε, ε_1, ε_2 its respective subproofs.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{T}_\omega\alpha \Rightarrow \mathsf{T}_\omega\alpha}\,(\mathrm{Ax}) \quad \dfrac{\dfrac{}{\alpha = 0 \Rightarrow \alpha = 0}\,(\mathrm{Ax})}{\alpha = 0 \Rightarrow \mathsf{cd}_{i+1}(\alpha, \beta')}\,(\vee\mathrm{R}) \quad \dots \quad \dfrac{\dfrac{}{\alpha = \overline{i-1} \Rightarrow \alpha = \overline{i-1}}\,(\mathrm{Ax})}{\alpha = \overline{i-1} \Rightarrow \mathsf{cd}_{i+1}(\alpha, \beta')}\,(\vee\mathrm{R}) \quad \varepsilon}{\mathsf{T}_\omega\beta \to \mathsf{cd}_0(\beta, \beta'), \mathsf{cd}_i(\alpha, \beta) \Rightarrow \mathsf{cd}_{i+1}(\alpha, \beta')}\,(\vee\mathrm{L})}{\mathsf{T}_\omega\beta \to \mathsf{cd}_0(\beta, \beta'), \mathsf{T}_\omega\alpha \to \mathsf{cd}_i(\alpha, \beta) \Rightarrow \mathsf{T}_\omega\alpha \to \mathsf{cd}_{i+1}(\alpha, \beta')}\,(\to\mathrm{L}, \to\mathrm{R})}{\mathsf{T}_\omega\beta \to \mathsf{cd}_0(\beta, \beta'), \mathsf{T}_\omega\alpha \to \mathsf{cd}_i(\alpha, \beta) \Rightarrow \mathsf{case}_{i+1}(\alpha)}\,(\forall\mathrm{L})}{\mathsf{Case}_0, \mathsf{case}_i(\alpha) \Rightarrow \mathsf{case}_{i+1}(\alpha)}\,(\exists\mathrm{L}, \forall\mathrm{L})}{\mathsf{Case}_0, \mathsf{Case}_i \Rightarrow \mathsf{Case}_{i+1}}\,(\forall\mathrm{R}, \forall\mathrm{L})$$

$$\dfrac{\dfrac{}{\mathsf{T}_\omega\beta \Rightarrow \mathsf{T}_\omega\beta}\,(\mathrm{Ax}) \quad \dfrac{\dfrac{\dfrac{}{\alpha = \mathsf{s}^i0 \Rightarrow \alpha = \mathsf{s}^i0}\,(\mathrm{Ax})}{\beta = 0, \alpha = \mathsf{s}^i\beta \Rightarrow \alpha = \mathsf{s}^i0}\,(=\mathrm{L}) \quad \varepsilon_2}{\mathsf{cd}_0(\beta, \beta'), \alpha = \mathsf{s}^i\beta \Rightarrow \alpha = \mathsf{s}^i0, \mathsf{T}_\omega\beta' \wedge \alpha = s^{i+1}\beta'}\,(\vee\mathrm{R})}{\mathsf{T}_\omega\beta \to \mathsf{cd}_0(\beta, \beta'), \mathsf{T}_\omega\beta \wedge \alpha = \mathsf{s}^i\beta \Rightarrow \mathsf{cd}_{i+1}(\alpha, \beta')}\,(\wedge\mathrm{L}, \to\mathrm{L})$$

$$\dfrac{\dfrac{}{\mathsf{T}_\omega\beta' \Rightarrow \mathsf{T}_\omega\beta'}\,(\mathrm{Ax}) \quad \dfrac{\dfrac{}{\alpha = \mathsf{s}^{i+1}\beta' \Rightarrow \alpha = s^{i+1}\beta'}\,(\mathrm{Ax})}{\beta = \mathsf{s}\beta', \alpha = \mathsf{s}^i\beta \Rightarrow \alpha = s^{i+1}\beta'}\,(=\mathrm{L})}{\mathsf{T}_\omega\beta' \wedge \beta = \mathsf{s}\beta', \alpha = \mathsf{s}^i\beta \Rightarrow \mathsf{T}_\omega\beta' \wedge \alpha = s^{i+1}\beta'}\,(\wedge\mathrm{R})$$

We prove the sequent $\mathsf{case}_0 \Rightarrow \mathsf{case}_i$ as follows.

$$\dfrac{\begin{array}{c}\vdots\pi\\ \mathsf{case}_0 \Rightarrow \mathsf{case}_{i-1}\end{array} \qquad \begin{array}{c}\vdots\gamma\\ \mathsf{case}_0, \mathsf{case}_{i-1} \Rightarrow \mathsf{case}_i\end{array}}{\mathsf{case}_0 \Rightarrow \mathsf{case}_i}\,(\mathrm{Cut})$$

□

Lemma 6.26. *Let $i \in \mathbb{N}$, then the sequent $\mathsf{T}_\omega\eta, \lceil\eta \not\approx 0\rceil, \dots, \lceil\eta \not\approx i-1\rceil, \lceil\eta \prec i\rceil \Rightarrow$ is provable in* LKID *with induction invariant $\mathsf{T}_\omega x \wedge \mathsf{case}_1(x)$.*

Proof. First we prove the sequent

$$\mathsf{case}_i(\eta), \mathsf{T}_\omega\eta, \lceil \eta \not\approx 0\rceil, \ldots, \lceil \eta \not\approx i-1\rceil, \lceil \eta \not\approx i\rceil \Rightarrow$$

in LKID-I as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\eta = 0 \Rightarrow \eta = 0}\,(\mathrm{Ax}) \quad \ldots \quad \dfrac{}{\eta = \overline{i-1} \Rightarrow \eta = \overline{i-1}}\,(\mathrm{Ax})\ \varepsilon}{\mathsf{cd}_1(\eta,\alpha), \Rightarrow \eta = 0, \ldots, \eta = i-1, \mathsf{T}_\omega\alpha \wedge \eta = \mathsf{s}^i\alpha}\,(\vee\mathrm{R}) \quad \dfrac{}{\mathsf{T}_\omega\eta \Rightarrow \mathsf{T}_\omega\eta}\,(\mathrm{Ax})}{\mathsf{cd}_1(\eta,\alpha), \neg(\mathsf{T}_\omega\alpha \wedge \eta = \mathsf{s}^i\alpha) \Rightarrow \eta = 0, \ldots, \eta = i-1}\,(\neg\mathrm{R})}{\mathsf{T}_\omega\eta \to \mathsf{cd}_1(\eta,\alpha), \mathsf{T}_\omega\eta, \neg(\mathsf{T}_\omega\alpha \wedge \eta = \mathsf{s}^i\alpha) \Rightarrow \eta = 0, \ldots, \eta = i-1}\,(\to\mathrm{R})}{\mathsf{T}_\omega\eta \to \mathsf{cd}_1(\eta,\alpha), \mathsf{T}_\omega\eta, \lceil \eta \prec i\rceil \Rightarrow \eta = 0, \ldots, \eta = i-1}\,(\forall\mathrm{R})}{\mathsf{T}_\omega\eta \to \mathsf{cd}_1(\eta,\alpha), \mathsf{T}_\omega\eta, \lceil \eta \not\approx 0\rceil, \ldots, \lceil \eta \not\approx i-1\rceil, \lceil \eta \prec i\rceil \Rightarrow .}\,(\neg\mathrm{R})}{\mathsf{case}_1(\eta), \mathsf{T}_\omega\eta, \lceil \eta \not\approx 0\rceil, \ldots, \lceil \eta \not\approx i-1\rceil, \lceil \eta \prec i\rceil \Rightarrow .}\,(\exists\mathrm{R})$$

The proof ε is given by

$$\dfrac{\dfrac{}{\mathsf{T}_\omega\alpha \Rightarrow \mathsf{T}_\omega\alpha}\,(\mathrm{Ax}) \quad \dfrac{}{\mathsf{s}^i\alpha = \eta \Rightarrow \mathsf{s}^i\alpha = \eta}\,(\mathrm{Ax})}{\mathsf{T}_\omega\beta \wedge \mathsf{s}^i\beta = \eta \Rightarrow \mathsf{T}_\omega\beta \wedge \eta = \mathsf{s}^i\beta}\,{\to}\mathrm{:L},\ {\wedge}\mathrm{:L}$$

By Lemmas 6.24, 6.25 the sequent $\mathsf{T}_\omega\eta, \lceil \eta \not\approx 0\rceil, \ldots, \lceil \eta \not\approx i-1\rceil, \lceil \eta \prec i\rceil \Rightarrow$ is provable in LKID with induction hypothesis $\mathsf{T}_\omega x \wedge \mathsf{case}^1(x)$ □

We can now assemble the previous result in order to obtain the main theorem of this thesis.

Theorem 6.27. *Let S be an L-clause set, δ an inference relation for S and $\sqsupseteq$ an immediate entailment relation. If S admits a cyclic superposition refutation w.r.t. δ and $\sqsupseteq$, involving the cycle $(i, j, S_{\mathsf{init}}, S_{\mathsf{loop}})$, then the sequent*

$$\mathrm{I}_\eta\lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i\eta), \mathrm{I}_x(\mathsf{T}_\omega x \wedge \mathsf{case}_1(x)), \mathrm{T}^{\mathsf{inj}}, \mathsf{T}_\omega\eta, \lceil S\rceil \Rightarrow$$

is provable in LKID-I.

Proof. By the definition of a cyclic superposition refutations, we have

$$S \vdash_\delta \eta \not\approx k, \text{ for all } k \in \{0, \ldots, i-1\}.$$

Hence by Lemma 6.13 there exist LKID-I proofs $\varepsilon_1, \ldots, \varepsilon_{i-1}$ of sequents

$$\mathrm{T}^{\mathsf{inj}}, \lceil S\rceil \Rightarrow \lceil \eta \not\approx k\rceil, \text{with } k \in \{0, \ldots, i-1\}.$$

Furthermore, by Lemma 6.14 there is an LKID-I proof γ of the sequent

$$\mathrm{T}^{\mathsf{inj}}, \lceil S\rceil \Rightarrow \lceil \eta \prec i\rceil.$$

Moreover, the proof γ only uses the induction invariant $\neg\lceil S_{\mathsf{init}}\rceil(s^i\eta)$. By Lemma 6.26 there exists an LKID-I proof π of sequent

$$\mathsf{T}_\omega\eta, \lceil \eta \not\approx 0\rceil, \ldots, \lceil \eta \not\approx i-1\rceil, \lceil \eta \prec i\rceil \Rightarrow .$$

The proof π only uses the induction invariant $\mathsf{T}_\omega x \wedge \mathsf{case}_1(x)$. Therefore the following proof only uses the induction invariants $\neg\lceil S_{\mathsf{init}}\rceil(s^i\eta)$ and $\mathsf{T}_\omega x \wedge \mathsf{case}_1(x)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots\varepsilon_0 \qquad \vdots\pi}{\mathrm{T}^{\mathsf{inj}}, \lceil S\rceil \Rightarrow \lceil \eta \not\approx 0\rceil \qquad \mathsf{T}_\omega\eta, \lceil \eta \not\approx 0\rceil, \ldots, \lceil \eta \not\approx i-1\rceil, \lceil \eta \prec i\rceil \Rightarrow}{\mathrm{T}^{\mathsf{inj}}, \mathsf{T}_\omega\eta, \lceil \eta \not\approx 1\rceil, \ldots, \lceil \eta \not\approx i-1\rceil, \lceil \eta \prec i\rceil\lceil S\rceil \Rightarrow}\,(\text{Cut}) \quad \vdots}{\dfrac{\varepsilon_{i-1} \qquad \mathrm{T}^{\mathsf{inj}}, \mathsf{T}_\omega\eta, \lceil \eta \not\approx i-1\rceil, \lceil \eta \prec i\rceil, \lceil S\rceil \Rightarrow}{\mathrm{T}^{\mathsf{inj}}, \mathsf{T}_\omega\eta, \lceil \eta \prec i\rceil, \lceil S\rceil \Rightarrow}\,(\text{Cut})}\,(\text{Cut})}{\mathrm{T}^{\mathsf{inj}}, \mathsf{T}_\omega\eta, \lceil S\rceil \Rightarrow}\,(\text{Cut})}$$

□

Corollary 6.28. *Let S be a clause set, δ an inference relation, and $\sqsupseteq$ an immediate entailment relation. If S admits a cyclic superposition refutation with respect to δ and $\sqsupseteq$, then the sequent $\mathrm{T}^{\mathsf{inj}}, \mathsf{T}_\omega\eta, \lceil S\rceil \Rightarrow$ is provable in $\mathsf{LKID}(\Sigma_1)$.*

Proof. Let $(i, j, S_{\mathsf{init}}, S_{\mathsf{loop}})$ be the inductive cycle involved in the refutation. For any clause $\mathcal{C} \in S_{\mathsf{init}}$, the formula $\lceil\mathcal{C}\rceil$ clearly is a Π_1-formula. Hence, there exists a Π_1-formula F, such that $F \leftrightarrow \lceil S_{\mathsf{init}}\rceil$ is provable in LKID-I. Therefore, there also exists a Σ_1-formula F' such that $F' \leftrightarrow \neg\lceil S_{\mathsf{init}}\rceil(\mathsf{s}^i\eta)$ is provable in LKID-I. The formula $\mathsf{T}_\omega x \wedge \mathsf{case}_1(x)$ clearly is equivalent in LKID-I to a Σ_1-formula G. Hence by Theorem 6.27 the sequent $\mathrm{T}^{\mathsf{inj}}, \mathsf{T}_\omega\eta, \lceil S\rceil \Rightarrow$ is provable with the Σ_1-induction invariants F' and G. □

This theorem allows us to confirm the informal conjecture that the n-clause calculus captures only a "weak" notion of induction. Although, it may be surprising that the n-clause calculus perhaps captures more than quantifier-free induction.

Moreover, Theorem 6.27 tells us that the cyclic superposition refutations of the n-clause calculus do only contain arguments by structural induction and not a stronger notion of induction, as is the case for the arguments encoded by the cyclic sequent calculus. In other words, we obtain a somewhat more fine-grained soundness for the n-clause calculus.

Corollary 6.29. *Let S be a clause set, δ an inference relation, and $\sqsupseteq$ an immediate entailment relation. If S admits a cyclic superposition refutation with respect to δ and $\sqsupseteq$, then $\mathrm{T}^{\mathsf{inj}}, \mathsf{T}_\omega\eta, \lceil S\rceil \Rightarrow$ is Henkin-valid.*

Proof. This corollary is an immediate consequence of Theorem 6.27, and the Henkin-soundness of the proof system LKID. □

There are two natural questions arising from Theorem 6.27. First of all it seems natural to ask whether our upper bound is optimal, that is, to ask whether the n-clause calculus actually proves a sentence which cannot be proved with quantifier-free induction in the system LKID. Furthermore it seems natural to ask whether the n-clause calculus is complete with respect to this notion of Σ_1-induction. In Chapter 7 we will give a little bit more detailed discussion of these new questions.

CHAPTER 7

Open Questions

In the previous sections we have shown that a refutation of a clause set S in the n-clause calculus can be simulated by $\mathsf{LKID}(\Sigma_1)$. From this observation two natural questions arise. First of all we might ask whether the induction invariants obtained by the translation outlined in Chapter 5 and Chapter 6 produces induction invariants that are optimal in their quantifier complexity. In other words, are there any clause sets whose translation cannot be proved unsatisfiable with quantifier-free induction? Secondly, we might also ask whether the n-clause calculus is complete with respect to Σ_1-induction. In Section 7.1 we will briefly discuss the optimiality of the obtained induction invariants. Section 7.2 gives some intuitions about the (in)completeness of the n-clause calculus with respect to Σ_1-induction.

7.1 Optimality of the Induction Invariants

The translation of the n-clause calculus to the to the calculus LKID via the calculus CLKID^ω was designed in such a way that it does not deviate to much from the underlying intuition of the n-clause calculus. It is apparently not the case that any unnecessary quantifiers are introduced. Indeed the only quantifiers that appear in the translation of clause sets are the quantifiers which are implied by the semantics of the n-clause logic. Hence, it would be surprising if it were the case that every refutable clause set could be refuted with a quantifier-free induction. Therefore it seems reasonable to start with the conjecture that there are indeed clause sets refutable by the n-clause calculus, which are not refuatble with quantifier-free induction.

In order to understand when and why a quantifier may be unnecessary, we start with an example where the translation yields an overly complicated induction hypothesis.

Consider the following clauses over the signature $\{\mathsf{t} : \iota, \mathsf{q} : \omega \to \iota\}$

$$\mathsf{q}(\mathsf{0}) \simeq \mathsf{t}, \tag{E4}$$

$$\mathsf{q}(x) \not\simeq \mathsf{t} \vee \mathsf{q}(\mathsf{s}x) \simeq \mathsf{t}, \tag{E5}$$

$$[\mathsf{q}(x) \not\simeq \mathsf{t} \mid \eta \simeq x]. \tag{E6}$$

It is easily seen that the clause $(0, 1, \{(\text{E6})\}, \{(\text{E6})\downarrow_1\})$ is an inductive cycle for this clause set. By Theorem 6.27 we obtain the induction invariant

$$\neg\forall x(\mathsf{T}_\omega x \wedge \eta = x \to \mathsf{q}(x) \neq \mathsf{t}).$$

In presence of the axiom $\mathsf{T}_\omega\eta$ the induction invariant is equivalent to the quantifier-free formula $\mathsf{q}(\eta) = \mathsf{t}$. Similarly, it is possible to eliminate one universal quantifier whenever the cycle's offset is 0. This quantifier was introduced because the inductive cycle represents an argument by infinite descent that starts with an assumption and descends along the chain of natural numbers, thus creating smaller counterexamples. Since the n-clause logic does not dispose of a symbol for the predecessor function, we need to use quantifiers.

In this example it turns out that the argumentation represented by the inductive cycle is not a very natural way to refute this clause set. A more natural way to prove refute this clause set is as follows. Observe that the clauses (E4) and (E5) are the actual inductive clause set. That is, (E4) and (E5) imply the clause $\mathsf{q}(x) \simeq \mathsf{t}$. The constraint clause (E6) provides some information contradicting the clause $\mathsf{q}(x) \simeq \mathsf{t}$. Hence, the clause set is unsatisfiable.

Let us now have a look at a similar clause set. This time, however, it is not so easy to obtain a quantifier-free induction invariant. Consider again the clauses (E1), (E2), and (E3) of Example 2.

$$\mathsf{p}(0, \mathsf{t}) \simeq \mathsf{t}, \tag{E1}$$

$$\mathsf{p}(x, y) \not\simeq \mathsf{t} \vee \mathsf{p}(sx, \mathsf{g}y) \simeq \mathsf{t}, \tag{E2}$$

$$[\mathsf{p}(x, y) \not\simeq \mathsf{t} \mid \eta \simeq x]. \tag{E3}$$

On the metalevel we have shown that this clause set is unsatisfiable by using a quantifier-free induction. But remember that this was only possible because our metalanguage allows us to explicitly express the n-fold iteration of a unary function symbol. Furthermore we have shown that $(0, 1, \{(\text{E3})\}, \{(\text{E3})\}\downarrow_1)$ is an inductive cycle. By Theorem 6.27 we obtain the formal induction invariant

$$\neg\forall x\forall y(\mathsf{T}_\omega x \wedge \mathsf{T}_\omega y \wedge \eta = x \to \mathsf{p}(x, y) \neq \mathsf{t}).$$

The quantifier binding the variable x is clearly superfluous and we have the equivalent induction hypothesis

$$\exists y(T_\omega y \wedge \mathsf{p}(\eta, y) = \mathsf{t}).$$

The existential quantifier serves to handle the absence of a construct allowing us to express explicitly the actual witness which is given by iterating the function g on t

depending on the value η. Therefore, it seems improbable that there exists a quantifier-free induction invariant which allows us to refute this clause set. Of course there is the possibility that there is a an inductive cycle involving only one variable. However, by looking at the clauses, and their deductive hull, it seems apparent that there is no such inductive cycle.

7.2 Incompleteness with respect to Σ_1-induction

If the n-clause calculus captures Σ_1-induction, it is natural to ask whether it is complete for this kind of induction. In the following we will consider two points, which may be a sources of incompleteness of the n-clause calculus with respect to Σ_1-induction.

7.2.1 Forward-Incompleteness of Superposition

A possible way to obtain a completeness result of the n-clause calculus with respect to Σ_1-induction would be to define the inverse translation to the translation defined in Chapters 5 and 6. That is, we need to convert Σ_1-induction invariants into clause sets that represent inductive cycles. There are several points to consider in doing so. First of all we need to translate the induction invariants to adequate clause sets. Secondly, and this is the more significant problem, we need to infer the existence of derivations satisfying the conditions for inductive cycles. This is more difficult to realize in so far, as the superposition calculus (as well as the resolution calculus) is subject to forward-incompleteness in the following sense.

Lemma 7.1. *There exists L-clauses $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $\mathcal{C}_1 \models \mathcal{C}_2$ but $\mathcal{C}_1 \nvdash \mathcal{C}_2$.*

Proof. We shall see that there are multiple sources of forward-incompleteness.

- Clauses are not weakened:

$$\{t_1 \simeq t_2\} \nvdash \{t_1 \simeq t_2 \vee l\}.$$

- The language of a clause is not unnecessarily extended:

$$\{t_1 \simeq t_2\} \nvdash \{\mathsf{f}(t_1) \simeq \mathsf{f}(t_2)\}.$$

- Instances are not generated without unification:

$$\{\mathsf{f}(x) \simeq t_1\} \nvdash \{\mathsf{f}(t_2) \simeq t_1\}.$$

□

Forward-incompletness in the above sense is not a weakness of the superposition calculus, but quite the contrary. The superposition calculus, as well as the resolution calculus, are meant to derive a contradiction (i.e. the empty clause) from unsatisfiable

clause sets. The superposition calculus is refutationally complete. This means that if a clause set is unsatisfiable, then a refutation of this clause set exists. Theoretically, there is simply no need to be able to derive some redundant clauses. Theorem provers based on the superposition calculus are usually implemented as saturation procedures. These provers try exhaustively to derive new clauses until the empty clause is eventually found. During this process a very large number of clauses may be generated. In order to keep this search effective and efficient it is thus necessary to avoid to generate too much redundant clauses.

This is in contrast with calculi such as LK that focus on proving the valid entailements. It is thus questionable whether the choice of using a superposition calculus as the underlying deductive system for the discovery of inductive cycles is an adequate choice.

The above observation shows that it may be the case that the reverse translation from $\mathrm{FOL}_{\mathrm{ID}}$ to the n-clause calculus is not in general possible because of the forward-incompleteness property of the superposition calculus. However finding an example where the forward-incompletness disables the discovery of an induction invariant is harder than one may expect. This is partially due to the numerous restrictions on the syntax of constraint clauses.

7.2.2 Different Forms of Induction

Another place to look for an example of incompleteness of the n-clause calculus are the different approaches to induction. We have seen in Section 1.1, that besides structural induction, there are various other possibilities to express induction on the natural numbers. It might thus be interesting to see how the n-clause calculus deals with clause sets that are natural for a specific type of induction. This idea is motivated by the fact that inductive cycles do not have the same properties as inductive formulas. For natural numbers it is possible to normalize j-step induction and even (i, j)-induction to the usual weak-induction. With inductive cycles this is not the case.

Lemma 7.2. *There exits an n-clause language L, an L-clause set S, an immediate entailment relation $\sqsupseteq$ and an inference relation δ such that S admits an inductive cycle $(1, 1, S_{\mathsf{init}}, S_{\mathsf{loop}})$ but S does not admit an induction loop $(0, 1, S'_{\mathsf{init}}, S'_{\mathsf{loop}})$.*

Proof. Let $L = \{\mathsf{t} : \iota, \mathsf{p} : \omega \to \iota\}$, $\sqsupseteq$ the equality relation, δ the unrestricted inference relation, and let S be the clause set containg exactly the following clauses.

$$[\mathsf{p}(\mathsf{s0}) \simeq \mathsf{t} \mid \Diamond], \tag{1}$$

$$[\mathsf{p}(x) \not\simeq \mathsf{t} \vee \mathsf{p}(\mathsf{s}x) \simeq \mathsf{t} \mid \Diamond], \tag{2}$$

$$[\mathsf{p}(x) \not\simeq \mathsf{t} \mid \eta \simeq x]. \tag{3}$$

By straightforward applications of superposition and reflection to the clauses above we obtain:

$$[\Box \mid \eta \simeq \mathsf{s0}], \tag{4}$$

$$[\mathsf{p}(x) \not\simeq \mathsf{t} \mid \eta \simeq \mathsf{s}x]. \tag{5}$$

Hence $(1, 1, \{(3)\}, \{(3)\}\downarrow_1)$ is an induction loop for S. It remains to show that S does not admit any loop with offset 0 and step 1. To this end we will show that $S \not\models [\square \mid \eta = 0]$ by providing a suitable interpretation.

We define the relations $\equiv_\iota$ and $\equiv_\omega$ as follows.

$$\equiv_\iota = (\{\mathsf{t}\} \cup \{\mathsf{p}(\overline{n}) : n \in \mathbb{N}, n > 0\})^2 \cup \{(\mathsf{p}(\mathsf{0}), \mathsf{p}(\mathsf{0}))\},$$
$$\equiv_\omega = \{(\overline{n}, \overline{n}) : n \in \mathbb{N}\}.$$

We set $\equiv = \equiv_\iota \cup \equiv_\omega$. It is easily seen that the relation $\equiv$ is a congruence relation. Hence $\mathcal{I} = (0, \equiv)$ is an L-interpretation. It remains to show that $\mathcal{I} \models S$ and $\mathcal{I} \not\models [\square \mid \eta \simeq \mathsf{0}]$.

By definition of $\equiv$ we have $\mathsf{p}(\overline{1}) \in [\mathsf{t}]_\equiv$ and hence $\mathcal{I} \models [\mathsf{p}(\overline{1}) \simeq \top \mid \Diamond]$. Let σ be any well-typed ground subsitution with $dom(\sigma) = \{x\}$. There are two possible cases either $x\sigma = \mathsf{0}$ or $x\sigma = \mathsf{s}\overline{n}$ for some $n \in \mathbb{N}$. If $x\sigma = \mathsf{0}$, then we have $\mathcal{I} \models [\mathsf{p}(x) \not\simeq \mathsf{t} \vee \mathsf{p}(\mathsf{s}x) \simeq \mathsf{t} \mid \Diamond]\sigma$ since $\mathsf{p}(\mathsf{0}) \notin [\mathsf{t}]_\equiv$. If $x\sigma = \mathsf{s}\overline{n}$, then we have $\mathcal{I} \models [\mathsf{p}(x) \not\simeq \mathsf{t} \vee \mathsf{p}(\mathsf{s}x) \simeq \mathsf{t} \mid \Diamond]\sigma$ since $\mathsf{p}(\mathsf{s}\overline{n}) \in [\mathsf{t}]_\equiv$. Hence $\mathcal{I}$ is a model of clause (2). Furthermore we have $\mathcal{I} \models [\mathsf{p}(x) \simeq \mathsf{t} \mid \eta \simeq x]$ since $\mathsf{p}(\mathsf{0}) \notin [\mathsf{t}]_\equiv$. Therefore $\mathcal{I} \models S$. Since $\mathcal{I} \not\models \square$ and $\eta^\mathcal{I} = 0$ we have $\mathcal{I} \not\models [\square \mid \eta \simeq \mathsf{0}]$. Thus $S \not\models [\square \mid \eta \simeq \mathsf{0}]$ and hence there does not exist $S' \subseteq S$ such that $S' \models [\square \mid \eta \simeq \mathsf{0}]$. Therefore the clause set S does not admit a loop with offset 0 and step 1. $\square$

Polynomial Induction

This principle is interesting in the context of the n-clause calculus since it turns out to be much easier to formalize in this calculus than other types of induction. Because of the syntactical restrictions on n-clause signatures (cf. Definition 3.1), it is difficult or even impossible to express even simple concepts such as the commutativity of the addition. Moreover, it is not possible to express the natural well-ordering of the natural numbers since it is not possible to introduce Skolem symbols for quantifiers ranging over natural numbers. The principle of polynomial induction can be formulated as follows.

$$\varphi(0) \wedge \forall x(\varphi(x) \to \varphi(2x) \wedge \varphi(2x+1)) \to \forall x \varphi(x).$$

Let us discuss how one would show the correctness of this induction principle having access to the natural well-ordering $<$. It seems intuitive to proceed by induction on the usual well-ordering $<$. We start by assuming that $\varphi(0)$ holds and that for all $m \in \mathbb{N}$ if $\varphi(m)$ holds, then we also have $\varphi(2m)$ and $\varphi(2m+1)$. Let now n be any natural number. There are two cases to consider. If $n = 0$ we are done. If $n > 0$, then there exists a number $m < n$ such that either $n = 2m$ or $n = 2m + 1$. By the induction hypothesis we have $\varphi(m)$. Thus, we obtain $\varphi(2m)$ and $\varphi(2m+1)$, and hence $\varphi(n)$ holds in both cases. Therefore, $\varphi(n)$ is true for all $n \in \mathbb{N}$.

In the following we will formulate a clause set which follows the pattern of polynomial induction. Because the n-clause logic does not allow other function symbols than $\mathsf{0}$ and s with range ω it is necessary to represent the function $x \mapsto 2 \cdot x$ by its graph. Let the

clause set S^{P} consist of the clauses given below.

$$\mathcal{C}_1 = \mathsf{d}(0,0) \simeq \mathsf{t}$$
$$\mathcal{C}_2 = \mathsf{d}(x,y) \not\simeq \mathsf{t} \vee \mathsf{d}(\mathsf{s}x, \mathsf{ss}y) \simeq \mathsf{t}$$
$$\mathcal{C}_3 = \mathsf{p}(0) \simeq \mathsf{t}$$
$$\mathcal{C}_4 = \mathsf{p}(x) \not\simeq \mathsf{t} \vee \mathsf{d}(x,y) \not\simeq \mathsf{t} \vee \mathsf{p}(y) \simeq \mathsf{t}$$
$$\mathcal{C}_5 = \mathsf{p}(x) \not\simeq \mathsf{t} \vee \mathsf{d}(x,y) \not\simeq \mathsf{t} \vee \mathsf{p}(\mathsf{s}y) \simeq \mathsf{t}$$
$$\mathcal{C}_6 = [p(x) \not\simeq \mathsf{t} \mid \eta \simeq x]$$

Let us now argue that this clause set is indeed unsatisfiable. First of all we need to show that our axiomatization of the predicate d captures indeed the graph of the function $x \mapsto 2 \cdot x$.

Lemma 7.3. *Let $\mathcal{I}$ be a model of S^{P}, then $\mathcal{I} \models \mathsf{d}(\overline{n}, \overline{2n}) \simeq \mathsf{t}$ for all $n \in \mathbb{N}$.*

Proof. We proceed by simple induction on n. The case where $n = 0$ is trivial. If $n = m + 1$, then by the induction hypothesis we have $\mathcal{I} \models d(\overline{m}, \overline{2m}) \simeq \mathsf{t}$. Since $\mathcal{I} \models \mathcal{C}_2$ we have $\mathcal{I} \models \mathsf{d}(\overline{m+1}, \overline{2m+2})$. □

Lemma 7.4. *Let $\mathcal{I}$ be a model of S^{P}, then $\mathcal{I} \models p(\overline{n}) \simeq \mathsf{t}$ for all $n \in \mathbb{N}$.*

Proof. We proceed by polynomial induction on the number n. The case where $n = 0$ is trivial. If $n > 0$, then there exists a number $m < n$ such that either $n = 2m$ or $n = 2m + 1$. By the induction hypothesis we obtain $\mathcal{I} \models \overline{p}(\overline{m})$ and by Lemma 7.3 we obtain $\mathcal{I} \models \overline{p}(\overline{m}, \overline{2m})$. Since $\mathcal{I} \models \mathcal{C}_4$ and $\mathcal{I} \models \mathcal{C}_5$ we obtain in both cases $\mathcal{I} \models p(\overline{n})$. □

The unsatifiability is now easily shown.

Lemma 7.5. *The clause set S^{P} is unsatisfiable.*

Proof. We proceed by contradiction and assume that there exists a model $\mathcal{I} = (n, \equiv)$ of S^{P}. The in particular we have $\mathcal{I} \models \mathcal{C}_6$ and thus $\mathcal{I} \models p(\overline{n}) \not\simeq \mathsf{t}$. But by Lemma 7.4 we also have $\mathcal{I} \models p(\overline{n}) \simeq \mathsf{t}$. Contradiction! □

The are two observations that we can make here. First of we all have used two distinct inductions to establish the unsatisfiability of this clause set: One structural induction to prove the encoding of the function $x \mapsto 2 \cdot x$, and a polynomial induction to obtain the actual contradiction. Thus it might be interesting to investigate how the n-clause calculus deals with clause sets that are naturally proved by such "nested" inductions. Secondly, it seems likely that the deductive hull of S^{P} does not contain an inductive cycle which can be used in a cyclic refutation. This is because it seems necessary to have an invariant that expresses that for a certain element x, $\mathsf{p}(y)$ is true for all y that precede x. However, this cannot be expressed in this language.

Nevertheless, there is one problem with the example. It is not clear whether the translation of the clause set S^{P} is refutable in LKID. Again this problem is related to the language. We need the usual well-ordering on the natural numbers in order to derive

the principle of polynomial induction but the language does not provide us with such a symbol. The situation is somewhat similar to that of the 2-Hydra statement presented in Section 4.4. That said, the clause set S^{P}, even though it might possibly be unprovable in the n-clause calculus, may not a very useful example for the incompleteness of the n-clause calculus with respect to Σ_1-induction, since it is likely that S^{P} is unprovable in LKID as well.

CHAPTER 8

Conclusion

The goal of this thesis was to give an upper bound on the quantifier complexity of the induction invariants required to simulate the refutations of the n-clause calculus in LK.

In Chapter 5 we have introduced a translation from n-clause logic to first-order logic with inductive definitions. Moreover we have proved that this translation exhibits desirable properties such as the equivalence with respect to satisfiability and the preservation of validity in one direction. Furthermore we have conjectured that validity is indeed preserved in both directions.

In Chapter 6 we outline two translations. The first translation takes a inductive cycles of the n-clause calculus and translates these cycles into cyclic proofs. We have seen that the cyclic sequent calculus is a suitable calculus for the representation of inductive cycles in the sense that these cycles appear to be a natural concept. In the second step, we have translated a restricted type of cyclic proofs, namely simple cyclic proofs, into inductive proofs. We have seen that this is possible because the loop sequent behaves somewhat similarly to an induction invariant. Moreover, we have seen that the resulting induction hypothesis is always equivalent to a Σ_1-formula. Finally, we have shown that it is possible to simulate n-clause refutations by adding a case distinction axiom which allows to establish the contradiction. The overall result is that an n-clause refutation uses only Σ_1-induction. More formally we have shown the following theorem, which is the central result of this thesis.

Theorem. *Let S be a clause set. If S is refutable in the n-clause calculus, then the sequent $\mathrm{T}^{\mathsf{inj}}, \mathsf{T}_\omega\eta, \lceil S \rceil \Rightarrow$ is provable in $\mathsf{LKID}(\Sigma_1)$.*

Finally, in Chapter 7 we have discussed two questions that arise from this result. We have argued that the induction invariants are in general likely to be optimal with respect the the quantifier complexity Σ_1 because there is some clause set, that is refutable by the n-clause calculus and that is apparently not refutable with quantifier-free induction in the system LKID. Furthermore we have discussed whether the n-clause calculus might be complete with respect to Σ_1-induction. We have seen that there is some reason to

belive that it is not complete with respect to this type of induction. The intuition being three possible sources of incompleteness: the forward incompleteness of the superposition calculus, nested inductions, and complicated induction principles.

Bibliography

[BGP12] James Brotherston, Nikos Gorogiannis, and Rasmus Lerchedahl Petersen. A generic cyclic theorem prover. In *APLAS*, volume 12, pages 350–367. Springer, 2012.

[BKPU16] Jasmin C. Blanchette, Cezary Kaliszyk, Lawrence C. Paulson, and Josef Urban. Hammering towards QED. *Journal of Formalized Reasoning*, 9(1):101–148, 2016.

[BS10] James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, 21(6):1177–1216, 2010.

[BSVH+93] Alan Bundy, Andrew Stevens, Frank Van Harmelen, Andrew Ireland, and Alan Smaill. Rippling: A heuristic for guiding inductive proofs. *Artificial intelligence*, 62(2):185–253, 1993.

[BT17a] Stefano Berardi and Makoto Tatsuta. Classical system of Martin-Löf's inductive definitions is not equivalent to cyclic proof system. In *International Conference on Foundations of Software Science and Computation Structures*, pages 301–317. Springer, 2017.

[BT17b] Stefano Berardi and Makoto Tatsuta. Equivalence of inductive definitions and cyclic proofs under arithmetic. In *Logic in Computer Science (LICS), 2017 32nd Annual ACM/IEEE Symposium on*, pages 1–12. IEEE, 2017.

[Bus98] Samuel R. Buss. An introduction to proof theory. *Handbook of proof theory*, 137:1–78, 1998.

[EH15] Sebastian Eberhard and Stefan Hetzl. Inductive theorem proving based on tree grammars. *Annals of Pure and Applied Logic*, 166(6):665–700, 2015.

[KP82] Laurie Kirby and Jeff Paris. Accessible independence results for Peano arithmetic. *Bulletin of the London Mathematical Society*, 14(4):285–293, 1982.

[KP13] Abdelkader Kersani and Nicolas Peltier. Combining superposition and induction: A practical realization. In *International Symposium on Frontiers of Combining Systems*, pages 7–22. Springer, 2013.

[McC97] William McCune. Solution of the Robbins problem. *Journal of Automated Reasoning*, 19(3):263–276, 1997.

[Sim17] Alex Simpson. Cyclic arithmetic is equivalent to Peano arithmetic. In *International Conference on Foundations of Software Science and Computation Structures*, pages 283–300. Springer, 2017.

[WH17] Tin Lok Wong and Stefan Hetzl. Some observations on the logical foundations of inductive theorem proving. *Logical Methods in Computer Science*, 13, 2017.