

Predicate Logic with Sequence Variables and Sequence Function Symbols^{*}

Temur Kutsia and Bruno Buchberger

Research Institute for Symbolic Computation
Johannes Kepler University Linz
A-4040 Linz, Austria
{kutsia,buchberger}@risc.uni-linz.ac.at

Abstract. We describe an extension of first-order logic with sequence variables and sequence functions. We define syntax, semantics and inference system for the extension so that Completeness, Compactness, Löwenheim-Skolem, and Model Existence theorems remain valid. The obtained logic can be encoded as a special order-sorted first-order theory. We also define an inductive theory with sequence variables and formulate induction rules. The calculus forms a basis for the top-down systematic theory exploration paradigm.

1 Introduction

The goal of future mathematical knowledge management is the availability of significant parts of mathematical knowledge in computer-processable, verified, well-structured and semantically unambiguous form over the web and the possibility to easily expand, modify, and re-structure this knowledge according to specifications defined by the user. For this, mathematical knowledge has to be formulated in the frame of formal logics. Translation between presentations of mathematical knowledge with respect to different logics should be automatic. A natural standard for such logics is (any version of) predicate logic.

We believe that the goal can only be achieved by a systematic build-up of mathematics from scratch using systematic, flexible, algorithmic tools based on algorithmic formal logic (automated reasoning). By these tools, most of the work involved in building up well-structured and reusable mathematical knowledge should be automated or, at least, computer-assisted. We call the research field that enables this type of generation of large pieces of coherent mathematical knowledge “Computer-Supported Mathematical Theory Exploration”.

The systems and projects like ALF [29], AUTOMATH [16], COQ [2], ELF [33], HOL [20], IMPS [1], ISABELLE [32], LEGO [28], MIZAR [3], NUPRL [15], OMEGA [4], and others have been designed and used to formalize mathematical knowledge. THEOREMA [12] is one of such projects, which aims at constructing tools for computer-supported mathematical theory exploration. Since then, within the THEOREMA project, various approaches to bottom-up and top-down

^{*} Supported by the Austrian Science Foundation (FWF) under Project SFB F1302.

computer-supported mathematical theory exploration have been proposed and pursued with an emphasis on top-down methods. These approaches and first results are documented in various publications and reports (see, e.g., [9, 37, 10, 11]). The approaches are summarized in the “lazy thinking paradigm” for mathematical theory exploration introduced by the second author in [10].

In general, mathematical theory exploration requires higher-order logic. The version of predicate logic used in the case studies on theory exploration [37, 10, 11] is a higher-order predicate logic with sequence variables and sequence functions. However, the proofs in the case studies are done essentially on the first-order level. In this paper we restrict ourselves to the first-order fragment of predicate logic with sequence variables and sequence functions.

Sequence variables can be instantiated with finite sequences of terms. They add expressiveness and elegance to the language and have been used in various knowledge representation systems like KIF [18] or Common Logic [21]. ISABELLE [32] implements sequent calculi using sequence variables. In programming, the language of MATHEMATICA [38] successfully uses pattern matching that supports sequence variables and flexible arity function symbols (see [8] for more details). Sequence functions can be interpreted as multi-valued functions and have been used (under different names) in reasoning or programming systems, like, e.g., in SET-VAR [6] or RELFUN [7].

The following example shows how the property of a function being “orderless” can be easily defined using sequence variables: $f(\bar{x}, x, \bar{y}, y, \bar{z}) = f(\bar{x}, y, \bar{y}, x, \bar{z})$ specifies that the order of arguments in terms with the head f and any number of arguments does not matter. The letters with the overbar are sequence variables. Without them, we would need a permutation to express the same property. Definition of concatenation $\langle \bar{x} \rangle \asymp \langle \bar{y} \rangle = \langle \bar{x}, \bar{y} \rangle$ is another example of the expressiveness sequence variables.

Using sequence variables in programming helps to write elegant and short code, like, for instance, implementing bubble sort in a rule-based manner:

$$\begin{aligned} \text{sort}(\langle \bar{x}, x, \bar{y}, y, \bar{z} \rangle) &:= \text{sort}(\langle \bar{x}, y, \bar{y}, x, \bar{z} \rangle) \text{ if } x > y \\ \text{sort}(\langle \bar{x} \rangle) &:= \langle \bar{x} \rangle \end{aligned}$$

Bringing sequence functions in the language naturally allows Skolemization over sequence variables: Let x, y be individual variables, \bar{x} be a sequence variable, and p be a flexible arity predicate symbol. Then $\forall x \forall y \exists \bar{x} p(x, y, \bar{x})$ Skolemizes to $\forall x \forall y p(x, y, \bar{f}(x, y))$, where \bar{f} is a binary Skolem sequence function symbol. Another example, $\forall \bar{y} \exists \bar{x} p(\bar{y}, \bar{x})$, where \bar{y} is a sequence variable, after Skolemization introduces a flexible arity sequence function symbol \bar{g} : $\forall \bar{y} p(\bar{y}, \bar{g}(\bar{y}))$.

Although sequence variables and sequence functions appear in various applications, so far, to our knowledge, there was no formal treatment of full predicate logic with these constructs. (Few exceptions are [22], that considers logic with sequence variables without sequence functions, and [25], investigating equational theories, again with sequence variables, but without sequence functions.) In this paper we fill this gap, describing syntax, semantics and inference system for an extension of classical first-order logic with sequence variables and sequence

functions. Although, in general, the extension is not very complicated, there are some subtle points that have to be treated carefully. We decided to include the detailed proofs in the paper to make it self-contained, although in many cases these proofs are straightforward generalizations of their standard counterparts.

In the extended language we allow both individual and sequence variables/function symbols, where the function symbols can have fixed or flexible arity. We have also predicates of fixed or flexible arity, and can quantify over individual and sequence variables. It gives a simple and elegant language, which can be encoded as a special order-sorted first-order theory.

A natural intuition behind sequence terms is that they represent finite sequences of individual terms. We formalize this intuition using induction, and introduce several versions of induction rules. Inductive theories with sequence variables have some interesting properties that one normally can not observe in their standard counterparts: For instance, the Herbrand universe is not an inductive domain, and induction rules can be defined without using constructors.

The calculus \mathbf{G}^{\approx} that we introduce in this paper generalizes \mathbf{LK}^{\approx} calculus (\mathbf{LK}^{\approx} is an extension of Gentzen's \mathbf{LK} calculus [19] with equality), and possesses many nice proof-theoretic properties, including the extended version of Gödel's completeness theorem. Also, the counterparts of Löwenheim-Skolem, Compactness, Model Existence theorems and Consistency lemma hold. \mathbf{G}^{\approx} together with induction and cut rules forms the logical basis of the top-down theory exploration procedure [10].

The main results of this paper are the following: First, we give the first detailed description of predicate logic with sequence variables and sequence functions, clarifying the intuitive meaning and formal semantics of sequence variables that some researchers considered to be insufficiently explored (see, e.g. [13]). Second, we describe the logical foundations of the “theory exploration with lazy thinking” paradigm.

The contributions of the first author are defining syntax and semantics of languages with sequence variables and sequence functions, designing and proving the properties of the calculus \mathbf{G}^{\approx} , and showing relations between induction rules and intended models. The second author pointed out the importance of using sequence variables in symbolic computation (see [8]), introduced sequence variables and sequence functions in the THEOREMA system, defined various inference rules for them (including induction), and designed provers that use sequence variables.

The paper is organized as follows: In Section 2 we describe the syntax. Substitutions are introduced in Section 3. In Section 4 semantics is defined. The inference system \mathbf{G} is introduced in Section 5 and its soundness and completeness is shown in sections 6 and 7. The inference system \mathbf{G}^{\approx} , obtained from \mathbf{G} by adding the rules for equality, and its soundness and completeness is considered in sections 8, 9 and 10. In Section 11 we discuss advantages and disadvantages of the calculi \mathbf{G} and \mathbf{G}^{\approx} . Section 12 defines induction with sequence variables. In Section 13 we show how to encode predicate logic with sequence variables and sequence functions in a special order-sorted first-order theory.

2 Syntax

We consider an alphabet consisting of the following pairwise disjoint sets of symbols:

We consider an alphabet consisting of the following pairwise disjoint sets of symbols: individual variables \mathcal{V}_{Ind} , sequence variables \mathcal{V}_{Seq} , fixed arity individual function symbols $\mathcal{F}_{\text{Ind}}^{\text{Fix}}$, flexible arity individual function symbols $\mathcal{F}_{\text{Ind}}^{\text{Flex}}$, fixed arity sequence function symbols $\mathcal{F}_{\text{Seq}}^{\text{Fix}}$, flexible arity sequence function symbols $\mathcal{F}_{\text{Seq}}^{\text{Flex}}$, fixed arity predicate symbols \mathcal{P}^{Fix} , and flexible arity predicate symbols $\mathcal{P}^{\text{Flex}}$. Each set of variables is countable. Each set of function and predicate symbols is finite or countable. In addition, $\mathcal{P}^{\text{Fix}} \neq \emptyset$. Besides, there are connectives $\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow$, quantifiers \exists, \forall , parenthesis $(,)$ and comma $,$ in the alphabet.

We will use the following denotations: $\mathcal{V} := \mathcal{V}_{\text{Ind}} \cup \mathcal{V}_{\text{Seq}}$; $\mathcal{F}_{\text{Ind}} := \mathcal{F}_{\text{Ind}}^{\text{Fix}} \cup \mathcal{F}_{\text{Ind}}^{\text{Flex}}$; $\mathcal{F}_{\text{Seq}} := \mathcal{F}_{\text{Seq}}^{\text{Fix}} \cup \mathcal{F}_{\text{Seq}}^{\text{Flex}}$; $\mathcal{F}^{\text{Fix}} := \mathcal{F}_{\text{Ind}}^{\text{Fix}} \cup \mathcal{F}_{\text{Seq}}^{\text{Fix}}$; $\mathcal{F}^{\text{Flex}} := \mathcal{F}_{\text{Ind}}^{\text{Flex}} \cup \mathcal{F}_{\text{Seq}}^{\text{Flex}}$; $\mathcal{F} := \mathcal{F}^{\text{Fix}} \cup \mathcal{F}^{\text{Flex}}$; $\mathcal{P} := \mathcal{P}^{\text{Fix}} \cup \mathcal{P}^{\text{Flex}}$. The *arity* of $q \in \mathcal{F}^{\text{Fix}} \cup \mathcal{P}^{\text{Fix}}$ is denoted by $\text{Ar}(q)$. A function symbol $c \in \mathcal{F}^{\text{Fix}}$ is called a *constant* if $\text{Ar}(c) = 0$.

Variables, connectives, quantifiers, punctuation marks and the binary equality predicate \approx are called *logical symbols*. The symbols in the set $\mathcal{F} \cup \mathcal{P} \setminus \{\approx\}$ are called *non-logical symbols*. Non-logical symbols define a *language*. We denote by \mathcal{L}_{\approx} the language defined by $\mathcal{F} \cup \mathcal{P}$ and call it a *language with equality*. The language \mathcal{L} , defined by $\mathcal{F} \cup \mathcal{P} \setminus \{\approx\}$, is called a *language without equality*.

Definition 1. We define the notion of term over \mathcal{F} and \mathcal{V} :

1. If $t \in \mathcal{V}_{\text{Ind}}$ (resp. $t \in \mathcal{V}_{\text{Seq}}$), then t is an individual (resp. sequence) term.
2. If $f \in \mathcal{F}_{\text{Ind}}^{\text{Fix}}$ (resp. $f \in \mathcal{F}_{\text{Seq}}^{\text{Fix}}$), $\text{Ar}(f) = n$, $n \geq 0$, and t_1, \dots, t_n are individual terms, then $f(t_1, \dots, t_n)$ is an individual (resp. sequence) term.
3. If $f \in \mathcal{F}_{\text{Ind}}^{\text{Flex}}$ (resp. $f \in \mathcal{F}_{\text{Seq}}^{\text{Flex}}$) and t_1, \dots, t_n ($n \geq 0$) are individual or sequence terms, then $f(t_1, \dots, t_n)$ is an individual (resp. sequence) term.

A term is either an individual or a sequence term.

We denote by $\mathcal{T}_{\text{Ind}}(\mathcal{F}, \mathcal{V})$, $\mathcal{T}_{\text{Seq}}(\mathcal{F}, \mathcal{V})$ and $\mathcal{T}(\mathcal{F}, \mathcal{V})$ respectively the set of all individual terms, all sequence terms and all terms over \mathcal{F} and \mathcal{V} .

If not otherwise stated, the following symbols, with or without indices, are used as metavariables: x, y and z over individual variables; \bar{x}, \bar{y} and \bar{z} over sequence variables; u and v over (individual or sequence) variables; f, g and h over individual function symbols; \bar{f}, \bar{g} and \bar{h} over sequence function symbols; a, b and c over individual constants; \bar{a}, \bar{b} and \bar{c} over sequence constants.

Example 1. Let $f \in \mathcal{F}_{\text{Ind}}^{\text{Flex}}$, $\bar{f} \in \mathcal{F}_{\text{Seq}}^{\text{Flex}}$, $g \in \mathcal{F}_{\text{Ind}}^{\text{Fix}}$, and $\bar{g} \in \mathcal{F}_{\text{Seq}}^{\text{Fix}}$, $\text{Ar}(g) = 2$ and $\text{Ar}(\bar{g}) = 1$.

1. $f(\bar{x}, g(x, y)) \in \mathcal{T}_{\text{Ind}}(\mathcal{F}, \mathcal{V})$.
2. $f(\bar{x}, \bar{f}(x, \bar{x}, y)) \in \mathcal{T}_{\text{Ind}}(\mathcal{F}, \mathcal{V})$.
3. $\bar{f}(\bar{x}, f(x, \bar{x}, y)) \in \mathcal{T}_{\text{Seq}}(\mathcal{F}, \mathcal{V})$.
4. $\bar{g}(f(x, \bar{x}, y)) \in \mathcal{T}_{\text{Seq}}(\mathcal{F}, \mathcal{V})$.

5. $f(\bar{x}, g(\bar{x}, y)) \notin \mathcal{T}(\mathcal{F}, \mathcal{V})$, because \bar{x} occurs as an argument of g .
6. $f(\bar{x}, \bar{g}(\bar{x})) \notin \mathcal{T}(\mathcal{F}, \mathcal{V})$, because \bar{x} occurs as an argument of \bar{g} .
7. $f(\bar{x}, \bar{g}(x, y)) \notin \mathcal{T}(\mathcal{F}, \mathcal{V})$, because \bar{g} is unary.

Definition 2. We define the notion of atomic formula or, shortly, atom, over \mathcal{P} , \mathcal{F} and \mathcal{V} :

1. If $p \in \mathcal{P}^{\text{Fix}}$, $\text{Ar}(p) = n$ with $n \geq 0$, and $t_1, \dots, t_n \in \mathcal{T}_{\text{Ind}}(\mathcal{F}, \mathcal{V})$, then $p(t_1, \dots, t_n)$ is an atom.
2. If $p \in \mathcal{P}^{\text{Flex}}$ and $t_1, \dots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ with $n \geq 0$, then $p(t_1, \dots, t_n)$ is an atom.

We denote the set of atoms over \mathcal{P} , \mathcal{F} and \mathcal{V} by $\mathcal{A}(\mathcal{P}, \mathcal{F}, \mathcal{V})$.

The function symbol f is called the *head* of $f(t_1, \dots, t_n)$. We denote the head of a term $t \notin \mathcal{V}$ by $\text{Head}(t)$. The head of an atom is defined in the similar way.

Definition 3. The set of formulae $\mathcal{Fml}(\mathcal{P}, \mathcal{F}, \mathcal{V})$ over \mathcal{P} , \mathcal{F} and \mathcal{V} is defined as follows:

$$\begin{aligned} \mathcal{Fml}(\mathcal{P}, \mathcal{F}, \mathcal{V}) := & \mathcal{A}(\mathcal{P}, \mathcal{F}, \mathcal{V}) \cup \{\neg A \mid A \in \mathcal{Fml}(\mathcal{P}, \mathcal{F}, \mathcal{V})\} \\ & \cup \{A \circ B \mid A, B \in \mathcal{Fml}(\mathcal{P}, \mathcal{F}, \mathcal{V}), \circ \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}\} \\ & \cup \{Qv.A \mid A \in \mathcal{Fml}(\mathcal{P}, \mathcal{F}, \mathcal{V}), v \in \mathcal{V}, Q \in \{\exists, \forall\}\}. \end{aligned}$$

Free and bound variables of a formula are defined in the standard way. A formula F is called *sentence* if the set of free variables of F is empty.

3 Substitutions

Definition 4. A variable binding is either a pair $x \mapsto t$ where $t \in \mathcal{T}_{\text{Ind}}(\mathcal{F}, \mathcal{V})$ and $t \neq x$, or an expression $\bar{x} \mapsto \ulcorner t_1, \dots, t_n \urcorner^1$ where $n \geq 0$, for all $1 \leq i \leq n$ we have $t_i \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, and if $n = 1$ then $t_1 \neq \bar{x}$.

Definition 5. A substitution is a finite set of variable bindings

$$\{x_1 \mapsto t_1, \dots, x_n \mapsto t_n, \bar{x}_1 \mapsto \ulcorner s_1^1, \dots, s_{k_1}^1 \urcorner, \dots, \bar{x}_m \mapsto \ulcorner s_1^m, \dots, s_{k_m}^m \urcorner\},$$

where $n, m \geq 0$, $x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_m$ are distinct variables.

Remark 1. In [27] we consider a more general notion of substitution that allows bindings for sequence function symbols as well. That, in fact, treats sequence function symbols as second-order variables. However, for our purposes the notion of substitution defined above is sufficient.

Lower case Greek letters are used to denote substitutions. The empty substitution is denoted by ε .

¹ To improve the readability, we write sequences that bind sequence variables between \ulcorner and \urcorner .

Definition 6. The instance of a term s with respect to a substitution σ , denoted $s\sigma$, is defined recursively as follows:

1. $x\sigma = \begin{cases} t, & \text{if } x \mapsto t \in \sigma, \\ x, & \text{otherwise.} \end{cases}$
2. $\bar{x}\sigma = \begin{cases} t_1, \dots, t_n, & \text{if } \bar{x} \mapsto \ulcorner t_1, \dots, t_n \urcorner \in \sigma, n \geq 0, \\ \bar{x}, & \text{otherwise.} \end{cases}$
3. $f(t_1, \dots, t_n)\sigma = f(t_1\sigma, \dots, t_n\sigma)$.

Example 2. $f(x, \bar{x}, \bar{y})\{x \mapsto a, \bar{x} \mapsto \ulcorner \cdot \urcorner, \bar{y} \mapsto \ulcorner a, f(\bar{x}), b \urcorner\} = f(a, a, f(\bar{x}), b)$.

Definition 7. Let σ be a substitution.

1. The domain of σ is the set of variables $\mathcal{D}om(\sigma) = \{l \mid l\sigma \neq l\}$.
2. The codomain of σ is the set of terms $\mathcal{C}od(\sigma) = \{l\sigma \mid l \in \mathcal{D}om(\sigma)\}$.
3. The range of σ is the set of variables: $\mathcal{R}an(\sigma) = \mathcal{V}ar(\mathcal{C}od(\sigma))$.

Note that a codomain of a substitution is a set of terms, not a set consisting of terms and sequences of terms. For instance, $\mathcal{C}od(\{x \mapsto b, \bar{x} \mapsto \ulcorner a, a, b \urcorner\}) = \{a, b\}$.

Application of a substitution on a formula is defined in the standard way. We denote an application of σ on F by $F\sigma$.

Definition 8. A term t is free for a variable v in a formula F if either

1. F is an atom, or
2. $F = \neg A$ and t is free for v in A , or
3. $F = (A \circ B)$ and t is free for v in B and C , where $\circ \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}$, or
4. $F = \forall u A$ or $F = \exists u A$ and either (a) $v = u$, or (b) $v \neq u$, $u \notin \mathcal{V}ar(t)$ and t is free for v in A .

We assume that for any formula F and substitution σ , before applying σ on F all bound variables in F are renamed so that they do not occur in $\mathcal{R}an(\sigma)$. It guarantees that for all $v \in \mathcal{D}om(\sigma)$ the terms in $v\sigma$ are free for v in F .

4 Semantics

4.1 Structures

For a set S , we denote by S^n the set of all n -tuples over S :

$$S^n = \{\langle s_1, \dots, s_n \rangle \mid s_1 \in S, \dots, s_n \in S\}.$$

In particular, $S^0 = \{\langle \rangle\}$. By S^∞ we denote the set $\cup_{i \geq 0} S^i$.

Definition 9. A structure \mathfrak{S} for the language \mathcal{L}_\approx (or, in short, an \mathcal{L}_\approx -structure) is a pair $\langle \mathcal{D}, \mathcal{I} \rangle$, where:

- \mathcal{D} is a non-empty set, called a domain of \mathfrak{S} , that is a disjoint union of two sets, \mathcal{D}_{Ind} and \mathcal{D}_{Seq} , written $\mathcal{D} = \mathcal{D}_{\text{Ind}} \uplus \mathcal{D}_{\text{Seq}}$, where $\mathcal{D}_{\text{Ind}} \neq \emptyset$.

- \mathcal{I} is a mapping, called an interpretation that associates:
 - To every individual constant c in \mathcal{L}_{\approx} some element $c_{\mathcal{I}}$ of \mathcal{D}_{Ind} .
 - To every sequence constant \bar{c} in \mathcal{L}_{\approx} some element $\bar{c}_{\mathcal{I}}$ of \mathcal{D}^{∞} .
 - To every n -ary individual function symbol f in \mathcal{L}_{\approx} , with $n > 0$, some n -ary function $f_{\mathcal{I}} : \mathcal{D}_{\text{Ind}}^n \rightarrow \mathcal{D}_{\text{Ind}}$.
 - To every n -ary sequence function symbol \bar{f} in \mathcal{L}_{\approx} , with $n > 0$, some n -ary multi-valued function $\bar{f}_{\mathcal{I}} : \mathcal{D}_{\text{Ind}}^n \rightarrow \mathcal{D}^{\infty}$.
 - To every flexible arity individual function symbol f in \mathcal{L}_{\approx} , some flexible arity function $f_{\mathcal{I}} : \mathcal{D}^{\infty} \rightarrow \mathcal{D}_{\text{Ind}}$.
 - To every flexible arity sequence function symbol \bar{f} in \mathcal{L}_{\approx} , some flexible arity multi-valued function $\bar{f}_{\mathcal{I}} : \mathcal{D}^{\infty} \rightarrow \mathcal{D}^{\infty}$.
 - To every n -ary predicate symbol p in \mathcal{L}_{\approx} , with $n \geq 0$, some n -ary predicate $p_{\mathcal{I}} \subseteq \mathcal{D}_{\text{Ind}}^n$;
 - To every flexible arity predicate symbol p in \mathcal{L}_{\approx} some flexible arity predicate $p_{\mathcal{I}} \subseteq \mathcal{D}^{\infty}$;

Definition 10. Let $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ be an \mathcal{L}_{\approx} -structure. A state σ over \mathfrak{S} , denoted $\sigma^{\mathfrak{S}}$, is a mapping defined on variables as follows:

- For an individual variable x , $\sigma^{\mathfrak{S}}(x) \in \mathcal{D}_{\text{Ind}}$.
- For a sequence variable \bar{x} , $\sigma^{\mathfrak{S}}(\bar{x}) \in \mathcal{D}^{\infty}$.

Definition 11. Let $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ be an \mathcal{L}_{\approx} -structure and let σ be a state over \mathfrak{S} . A value of a term t in \mathfrak{S} with respect to σ , denoted $\text{Val}_{\sigma}^{\mathfrak{S}}(t)$, is defined as follows:

- $\text{Val}_{\sigma}^{\mathfrak{S}}(v) = \sigma^{\mathfrak{S}}(v)$, for every $v \in \mathcal{V}$.
- $\text{Val}_{\sigma}^{\mathfrak{S}}(f(t_1, \dots, t_n)) = f_{\mathcal{I}}(\text{Val}_{\sigma}^{\mathfrak{S}}(t_1), \dots, \text{Val}_{\sigma}^{\mathfrak{S}}(t_n))$, for every $f(t_1, \dots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, $n \geq 0$.

Definition 12. Let v be a variable and $\sigma^{\mathfrak{S}}$ be a state over an \mathcal{L}_{\approx} -structure \mathfrak{S} . We say that a state $\vartheta^{\mathfrak{S}}$ is a v -variant of $\sigma^{\mathfrak{S}}$ iff $\vartheta^{\mathfrak{S}}(u) = \sigma^{\mathfrak{S}}(u)$ for each variable $u \neq v$.

Definition 13. Let $\sigma^{\mathfrak{S}}$ be a state over an \mathcal{L}_{\approx} -structure $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$. Let also $d \in \mathcal{D}_{\text{Ind}}$ and $d_1, \dots, d_n \in \mathcal{D}$. Then

1. $\sigma^{\mathfrak{S}}[x := d]$ denotes a state $\vartheta^{\mathfrak{S}}$ such that $\vartheta^{\mathfrak{S}}$ is an x -variant of $\sigma^{\mathfrak{S}}$, and $\vartheta^{\mathfrak{S}}(x) = d$.
2. $\sigma^{\mathfrak{S}}[\bar{x} := \ulcorner d_1, \dots, d_n \urcorner]$ denotes a state $\vartheta^{\mathfrak{S}}$ such that $\vartheta^{\mathfrak{S}}$ is an \bar{x} -variant of $\sigma^{\mathfrak{S}}$, and $\vartheta^{\mathfrak{S}}(\bar{x}) = \ulcorner d_1, \dots, d_n \urcorner$.

The set $\mathcal{TV} = \{\mathbf{T}, \mathbf{F}\}$ is called the set of truth values. The operations $\neg_{\text{TV}}, \vee_{\text{TV}}, \wedge_{\text{TV}}, \Rightarrow_{\text{TV}}, \Leftrightarrow_{\text{TV}}$ are defined on \mathcal{TV} in the standard way

Definition 14. Let $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ be an \mathcal{L}_{\approx} -structure and σ be a state over \mathfrak{S} . A truth value of a formula F in \mathfrak{S} with respect to σ , denoted $\text{Val}_{\sigma}^{\mathfrak{S}}(F)$, is defined as follows:

- $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(p(t_1, \dots, t_n)) = \mathbf{T}$ iff $\langle \mathcal{V}al_{\sigma}^{\mathfrak{S}}(t_1), \dots, \mathcal{V}al_{\sigma}^{\mathfrak{S}}(t_n) \rangle \subseteq p_{\mathcal{I}}$.
- $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(t_1 \approx t_2) = \mathbf{T}$ iff $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(t_1) = \mathcal{V}al_{\sigma}^{\mathfrak{S}}(t_2)$.
- $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\neg F) = \neg_{\text{TV}} \mathcal{V}al_{\sigma}^{\mathfrak{S}}(F)$.
- $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(F_1 \circ F_2) = \mathcal{V}al_{\sigma}^{\mathfrak{S}}(F_1) \circ_{\text{TV}} \mathcal{V}al_{\sigma}^{\mathfrak{S}}(F_2)$, where $\circ \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}$
- $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\forall v F) = \mathbf{T}$ iff $\mathcal{V}al_{\vartheta}^{\mathfrak{S}}(F) = \mathbf{T}$ for every $\vartheta^{\mathfrak{S}}$ that is a v -variant of $\sigma^{\mathfrak{S}}$.
- $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\exists v F) = \mathbf{T}$ iff $\mathcal{V}al_{\vartheta}^{\mathfrak{S}}(F) = \mathbf{T}$ for some $\vartheta^{\mathfrak{S}}$ that is a v -variant of $\sigma^{\mathfrak{S}}$.

Definition 15. Let \mathfrak{S} be an \mathcal{L}_{\approx} -structure.

1. A formula F is true (resp. false) in \mathfrak{S} with respect to a state $\sigma^{\mathfrak{S}}$ iff $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(F) = \mathbf{T}$ (resp. $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(F) = \mathbf{F}$).
2. A formula F is satisfiable in \mathfrak{S} , iff $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(F) = \mathbf{T}$ for some state $\sigma^{\mathfrak{S}}$.
3. A formula F is true in \mathfrak{S} , denoted $\mathfrak{S} \models F$, iff $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(F) = \mathbf{T}$ for all state $\sigma^{\mathfrak{S}}$. In this case \mathfrak{S} is called a model of F .
4. A formula F is valid, denoted $\models F$, iff F is true in all structures for the language.
5. A set Γ of formulae is satisfiable in \mathfrak{S} iff there exists a state $\sigma^{\mathfrak{S}}$ such that $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(F) = \mathbf{T}$ for all $F \in \Gamma$.
6. A set Γ of formulae is satisfiable iff it is satisfiable in some \mathcal{L}_{\approx} -structure.
7. A structure \mathfrak{S} is a model of a set of formulae Γ , denoted $\mathfrak{S} \models \Gamma$, iff $\mathfrak{S} \models F$ for all $F \in \Gamma$.
8. A set Γ of formulae is valid, denoted $\models \Gamma$, iff $\mathfrak{S} \models \Gamma$ for every \mathcal{L}_{\approx} -structure \mathfrak{S} .
9. A formula F is a logical consequence of a set Γ of formulae, denoted $\Gamma \models F$, iff for every \mathcal{L} -structure \mathfrak{S} and for every state $\sigma^{\mathfrak{S}}$ over \mathfrak{S} , if $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(A) = \mathbf{T}$ for every formula $A \in \Gamma$, then $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(F) = \mathbf{T}$.

Definition 16. Let $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ be an \mathcal{L}_{\approx} -structure. The extended language $\mathcal{L}_{\approx}(\mathfrak{S})$ is obtained from \mathcal{L}_{\approx} by adding to \mathcal{F}^{Fix} a new individual constant $a_{\mathfrak{S}}$ for each $a \in \mathcal{D}_{\text{Ind}}$ and a new sequence constant $\bar{b}_{\mathfrak{S}}$ for each $\bar{b} \in \mathcal{D}_{\text{Seq}}$. The interpretation \mathcal{I} is extended to the new individual or sequence constants $d_{\mathfrak{S}}$ by defining $d_{\mathfrak{S}\mathcal{I}} = d$. The constant $d_{\mathfrak{S}}$ is called a name of $d \in \mathcal{D}$. Thus, for every state $\sigma^{\mathfrak{S}}$ over \mathfrak{S} , and for every name $d_{\mathfrak{S}}$ of an element $d \in \mathcal{D}$, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(d_{\mathfrak{S}}) = d$.

Lemma 1. Let $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ be an \mathcal{L}_{\approx} -structure, $\sigma^{\mathfrak{S}}$ be a state over \mathfrak{S} , and E be either term or formula. Then the following hold:

1. For any element $d \in \mathcal{D}_{\text{Ind}}$ and any individual variable x ,

$$\mathcal{V}al_{\sigma}^{\mathfrak{S}}[x:=d](E) = \mathcal{V}al_{\sigma}^{\mathfrak{S}}(E\{x \mapsto d_{\mathfrak{S}}\}).$$

2. For any elements $d_1, \dots, d_n \in \mathcal{D}$ and any sequence variable \bar{x} ,

$$\mathcal{V}al_{\sigma}^{\mathfrak{S}}[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner](E) = \mathcal{V}al_{\sigma}^{\mathfrak{S}}(E\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}).$$

Proof. The lemma can be proved by induction on terms and formulae. Here we give the proof of the part 2 only for E being a formula.

If E is an atom of the form $p(t_1, \dots, t_m)$, then

$$\begin{aligned}
\mathcal{V}al_{\sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner]}^{\mathfrak{S}}(p(t_1, \dots, t_m)) &= \mathbf{T} \text{ iff, by Definition 14,} \\
\langle \mathcal{V}al_{\sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner]}^{\mathfrak{S}}(t_1), \dots, \mathcal{V}al_{\sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner]}^{\mathfrak{S}}(t_m) \rangle &\subseteq P_{\mathcal{I}} \text{ iff,} \\
\langle \mathcal{V}al_{\sigma}^{\mathfrak{S}}(t_1\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}), \dots, \\
\mathcal{V}al_{\sigma}^{\mathfrak{S}}(t_m\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) \rangle &\subseteq P_{\mathcal{I}} \text{ iff, by Definition 14,} \\
\mathcal{V}al_{\sigma}^{\mathfrak{S}}(p(t_1, \dots, t_m)\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) &= \mathbf{T}.
\end{aligned}$$

If E is of the form $\neg A$, then

$$\begin{aligned}
\mathcal{V}al_{\sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner]}^{\mathfrak{S}}(\neg A) &= \neg_{\text{TV}} \mathcal{V}al_{\sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner]}^{\mathfrak{S}}(A) \\
&= \neg_{\text{TV}} \mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) \\
&= \mathcal{V}al_{\sigma}^{\mathfrak{S}}(\neg A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\})
\end{aligned}$$

If E is of the form $A \circ B$, where $\circ \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}$, then

$$\begin{aligned}
\mathcal{V}al_{\sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner]}^{\mathfrak{S}}(A \circ B) &= \mathcal{V}al_{\sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner]}^{\mathfrak{S}}(A) \circ_{\text{TV}} \mathcal{V}al_{\sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner]}^{\mathfrak{S}}(B) \\
&= \mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) \\
&\quad \circ_{\text{TV}} \mathcal{V}al_{\sigma}^{\mathfrak{S}}(B\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) \\
&= \mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\} \\
&\quad \circ B\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}).
\end{aligned}$$

If E is of the form $\forall v A$, there are three cases. If $v \in \mathcal{V}_{\text{Ind}}$, then

$$\begin{aligned}
\mathcal{V}al_{\sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner]}^{\mathfrak{S}}(\forall v A) &= \mathbf{T} \text{ iff, by Definition 14, for all } a \in \mathcal{D}_{\text{Ind}} \\
\mathcal{V}al_{\sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner][v:=a]}^{\mathfrak{S}}(A) &= \mathbf{T}.
\end{aligned}$$

By the induction hypothesis, we have

$$\begin{aligned}
\mathcal{V}al_{\sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner][v:=a]}^{\mathfrak{S}}(A) &= \mathcal{V}al_{\sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner]}^{\mathfrak{S}}(A\{v \mapsto a_{\mathfrak{S}}\}) \\
&= \mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{v \mapsto a_{\mathfrak{S}}\}\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\})
\end{aligned}$$

Since $a_{\mathfrak{S}}, d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}}$ are constants and $v \neq \bar{x}$,

$$\begin{aligned}
\mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{v \mapsto a_{\mathfrak{S}}\}\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) &= \\
\mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}\{v \mapsto a_{\mathfrak{S}}\}). &
\end{aligned}$$

Hence, by the induction hypothesis and Definition 14 we have

$$\begin{aligned}
\mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}\{v \mapsto a_{\mathfrak{S}}\}) &= \mathbf{T} \text{ for all } a \in \mathcal{D}_{\text{Ind}} \text{ iff} \\
\mathcal{V}al_{\sigma[v:=a]}^{\mathfrak{S}}(A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) &= \mathbf{T} \text{ for all } a \in \mathcal{D}_{\text{Ind}} \text{ iff} \\
\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\forall v A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) &= \mathbf{T}.
\end{aligned}$$

Therefore

$$\begin{aligned} \mathcal{V}al_{\sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner]}^{\mathfrak{S}}(\forall v A) &= \mathbf{T} \text{ iff} \\ \mathcal{V}al_{\sigma}^{\mathfrak{S}}(\forall v A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) &= \mathbf{T}. \end{aligned}$$

If $v \in \mathcal{V}_{\text{Seq}}$ and $v \neq \bar{x}$, we can proceed in the similar way as above, taking a sequence of elements $a_1, \dots, a_k \in \mathcal{D}$ instead of a single element from \mathcal{D}_{Ind} .

If $v = \bar{x}$, then we have

$$\begin{aligned} \sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner][v:=\ulcorner a_1, \dots, a_k \urcorner] &= \sigma[\bar{x}:=\ulcorner a_1, \dots, a_k \urcorner] \\ (\forall v A)\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\} &= \forall v A \end{aligned}$$

and so

$$\begin{aligned} \mathcal{V}al_{\sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner]}^{\mathfrak{S}}(\forall v A) &= \mathbf{T} \text{ iff} \\ \mathcal{V}al_{\sigma[\bar{x}:=\ulcorner d_1, \dots, d_n \urcorner][v:=\ulcorner a_1, \dots, a_k \urcorner]}^{\mathfrak{S}}(A) &= \mathbf{T} \text{ for all } a_1, \dots, a_k \in \mathcal{D} \text{ iff} \\ \mathcal{V}al_{\sigma[\bar{x}:=\ulcorner a_1, \dots, a_k \urcorner]}^{\mathfrak{S}}(A) &= \mathbf{T} \text{ for all } a_1, \dots, a_k \in \mathcal{D} \text{ iff} \\ \mathcal{V}al_{\sigma}^{\mathfrak{S}}(\forall \bar{x} A) &= \mathbf{T} \text{ iff} \\ \mathcal{V}al_{\sigma}^{\mathfrak{S}}((\forall v A)\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) &= \mathbf{T}. \end{aligned}$$

The case when E is of the form $\exists v A$ is similar to the previous case. \square

Using Lemma 1 it is easy to prove the lemma below that states that the new definition of the truth of quantified formula is equivalent to the old one.

Lemma 2. *Let $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ be an \mathcal{L} -structure. Then for any formula A , any state $\sigma^{\mathfrak{S}}$ over \mathfrak{S} , any individual variable x and any sequence variable \bar{x} , the following hold:*

1. $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\forall x A) = \mathbf{T}$ iff for all $d \in \mathcal{D}_{\text{Ind}}$,

$$\mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{x \mapsto d_{\mathfrak{S}}\}) = \mathbf{T}.$$

2. $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\exists x A) = \mathbf{T}$ iff for some $d \in \mathcal{D}_{\text{Ind}}$,

$$\mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{x \mapsto d_{\mathfrak{S}}\}) = \mathbf{T}.$$

3. $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\forall \bar{x} A) = \mathbf{T}$ iff for all $d_1, \dots, d_n \in \mathcal{D}$,

$$\mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) = \mathbf{T}.$$

4. $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\exists \bar{x} A) = \mathbf{T}$ iff for some $d_1, \dots, d_n \in \mathcal{D}$,

$$\mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) = \mathbf{T}.$$

The next result states that for any formula F and any state $\sigma^{\mathfrak{S}}$ over some \mathcal{L}_{\approx} -structure \mathfrak{S} , the truth value $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(F)$ only depends on the restriction of $\sigma^{\mathfrak{S}}$ to the set of free variables of A .

Lemma 3. Let F be a formula, \mathfrak{S} be an \mathcal{L}_{\approx} -structure, and $\sigma^{\mathfrak{S}}$ and $\vartheta^{\mathfrak{S}}$ be two states over \mathfrak{S} such that $\sigma^{\mathfrak{S}}(v) = \vartheta^{\mathfrak{S}}(v)$ for each variable v free in F . Then $\text{Val}_{\sigma^{\mathfrak{S}}}(F) = \text{Val}_{\vartheta^{\mathfrak{S}}}(F)$.

Proof. By induction on terms and formulae. \square

Corollary 1. Let F be a sentence. Then for every \mathcal{L}_{\approx} -structure \mathfrak{S} , either $\mathfrak{S} \models F$ or $\mathfrak{S} \models \neg F$.

4.2 Signed Formulae

Definition 17. Signed formulae of type α , type β , type γ and type δ , and their components are defined in the tables below.

Type- α formulae

α	α_1	α_2
$\text{T}(A \wedge B)$	$\text{T}A$	$\text{T}B$
$\text{F}(A \vee B)$	$\text{F}A$	$\text{F}B$
$\text{F}(A \Rightarrow B)$	$\text{T}A$	$\text{F}B$
$\text{T}(\neg A)$	$\text{F}A$	$\text{F}A$
$\text{F}(\neg A)$	$\text{T}A$	$\text{T}B$

Type- β formulae

β	β_1	β_2
$\text{F}(A \wedge B)$	$\text{F}A$	$\text{F}B$
$\text{T}(A \vee B)$	$\text{T}A$	$\text{T}B$
$\text{T}(A \Rightarrow B)$	$\text{F}A$	$\text{T}B$

Type- γ formulae

γ	γ_1	γ_2
$\text{T}(\forall x A)$	$\text{T}(A\{x \mapsto t\})$	$\text{T}(A\{x \mapsto t\})$
$\text{F}(\exists x A)$	$\text{F}(A\{x \mapsto t\})$	$\text{F}(A\{x \mapsto t\})$
$\text{T}(\forall \bar{x} A)$	$\text{T}(A\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\})$	$\text{T}(A\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\})$
$\text{F}(\exists \bar{x} A)$	$\text{F}(A\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\})$	$\text{F}(A\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\})$

where t is an individual term free for x in A , and s_1, \dots, s_n is a (possibly empty) sequence of terms free for \bar{x} in A .

Type- δ formulae

δ	δ_1	δ_2
$\text{T}(\exists x A)$	$\text{T}(A\{x \mapsto t\})$	$\text{T}(A\{x \mapsto t\})$
$\text{F}(\forall x A)$	$\text{F}(A\{x \mapsto t\})$	$\text{F}(A\{x \mapsto t\})$
$\text{T}(\exists \bar{x} A)$	$\text{T}(A\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\})$	$\text{T}(A\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\})$
$\text{F}(\forall \bar{x} A)$	$\text{F}(A\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\})$	$\text{F}(A\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\})$

where t is an individual term free for x in A , and s_1, \dots, s_n is a (possibly empty) sequence of terms free for \bar{x} in A .

Sometimes we distinguish two subtypes of type- γ and type- δ formulae: We call $\mathsf{T}(\forall xA)$ and $\mathsf{F}(\exists xA)$ formulae of type γ_i ; $\mathsf{T}(\forall \bar{x}A)$ and $\mathsf{F}(\exists \bar{x}A)$ formulae of type γ_s ; $\mathsf{T}(\exists xA)$ and $\mathsf{F}(\forall xA)$ formulae of type δ_i ; $\mathsf{T}(\exists \bar{x}A)$ and $\mathsf{F}(\forall \bar{x}A)$ formulae of type δ_s .

A formula $\mathsf{T}A$ (resp. $\mathsf{F}A$) is called a *conjugate* of a formula $\mathsf{F}A$ (resp. $\mathsf{T}A$).

For a formula C of type γ , we introduce the following notation:

$$C(t) = \begin{cases} \mathsf{T}(C\{x \mapsto t\}), & \text{if } C = \mathsf{T}(\forall xA) \\ \mathsf{F}(C\{x \mapsto t\}), & \text{if } C = \mathsf{F}(\exists xA) \end{cases}$$

$$C(s_1, \dots, s_n) = \begin{cases} \mathsf{T}(C\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\}), & \text{if } C = \mathsf{T}(\forall \bar{x}A) \\ \mathsf{F}(C\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\}), & \text{if } C = \mathsf{F}(\exists \bar{x}A) \end{cases}$$

For a formula D of type δ , we introduce the following notation:

$$D(t) = \begin{cases} \mathsf{T}(D\{x \mapsto t\}), & \text{if } D = \mathsf{T}(\exists xA) \\ \mathsf{F}(D\{x \mapsto t\}), & \text{if } D = \mathsf{F}(\exists xA) \end{cases}$$

$$D(s_1, \dots, s_n) = \begin{cases} \mathsf{T}(D\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\}), & \text{if } D = \mathsf{T}(\exists \bar{x}A) \\ \mathsf{F}(D\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\}), & \text{if } D = \mathsf{F}(\forall \bar{x}A) \end{cases}$$

We define semantics of signed formulae as follows.

Definition 18. Given an \mathcal{L}_{\approx} -structure \mathfrak{S} , a state $\sigma^{\mathfrak{S}}$, and a formula A ,

1. $\mathcal{V}al_{\sigma^{\mathfrak{S}}}^{\mathfrak{S}}(\mathsf{T}A) = \mathbf{T}$ iff $\mathcal{V}al_{\sigma^{\mathfrak{S}}}^{\mathfrak{S}}(A) = \mathbf{T}$.
2. $\mathcal{V}al_{\sigma^{\mathfrak{S}}}^{\mathfrak{S}}(\mathsf{F}A) = \mathbf{T}$ iff $\mathcal{V}al_{\sigma^{\mathfrak{S}}}^{\mathfrak{S}}(\neg A) = \mathbf{T}$.

The following lemma follows from this definition and Lemma 1.

Lemma 4. For any \mathcal{L}_{\approx} -structure $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ and any state $\sigma^{\mathfrak{S}}$:

1. For any formula C of type γ_i , $\mathcal{V}al_{\sigma^{\mathfrak{S}}}^{\mathfrak{S}}(C) = \mathbf{T}$ iff $\mathcal{V}al_{\sigma^{\mathfrak{S}}}^{\mathfrak{S}}(C(d_{\mathfrak{S}})) = \mathbf{T}$ for every $d \in \mathcal{D}_{\text{Ind}}$.
2. For any formula C of type γ_s , $\mathcal{V}al_{\sigma^{\mathfrak{S}}}^{\mathfrak{S}}(C) = \mathbf{T}$ iff $\mathcal{V}al_{\sigma^{\mathfrak{S}}}^{\mathfrak{S}}(C(d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}})) = \mathbf{T}$ for every $d_1, \dots, d_n \in \mathcal{D}$.
3. For any formula D of type δ_i , $\mathcal{V}al_{\sigma^{\mathfrak{S}}}^{\mathfrak{S}}(D) = \mathbf{T}$ iff $\mathcal{V}al_{\sigma^{\mathfrak{S}}}^{\mathfrak{S}}(D(d_{\mathfrak{S}})) = \mathbf{T}$ for some $d \in \mathcal{D}_{\text{Ind}}$.
4. For any formula D of type δ_s , $\mathcal{V}al_{\sigma^{\mathfrak{S}}}^{\mathfrak{S}}(D) = \mathbf{T}$ iff $\mathcal{V}al_{\sigma^{\mathfrak{S}}}^{\mathfrak{S}}(D(d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}})) = \mathbf{T}$ for some $d_1, \dots, d_n \in \mathcal{D}$.

We call formulae of type α formulae of *conjunctive type*, formulae of type β formulae of *disjunctive type*, formulae of type γ formulae of *universal type*, and formulae of type δ are called formulae of *existential type*.

Definition 19. Given a language \mathcal{L}_{\approx} , a nonempty set $\mathcal{A} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{V})$ of terms of \mathcal{L}_{\approx} is called a term algebra iff:

1. For every $f \in \mathcal{F}^{\text{Fix}}$ with $\text{Ar}(f) = n$, $n \geq 0$, and for any individual terms t_1, \dots, t_n in \mathcal{A} , the term $f(t_1, \dots, t_n)$ is in \mathcal{A} .

2. For every $f \in \mathcal{F}^{\text{Flex}}$ and for any terms t_1, \dots, t_n in \mathcal{A} , the term $f(t_1, \dots, t_n)$ is in \mathcal{A} .

Any term algebra \mathcal{A} can be represented as a disjoint union $\mathcal{A} = \mathcal{A}_{\text{Ind}} \uplus \mathcal{A}_{\text{Seq}}$, where $\mathcal{A}_{\text{Ind}} = \mathcal{A} \cap \mathcal{T}_{\text{Ind}}(\mathcal{F}, \mathcal{V})$ and $\mathcal{A}_{\text{Seq}} = \mathcal{A} \cap \mathcal{T}_{\text{Seq}}(\mathcal{F}, \mathcal{V})$.

Definition 20. Let \mathcal{L} and \mathcal{L}' be two languages (with or without equality).

1. \mathcal{L}' is called a reduct of \mathcal{L} and \mathcal{L} is called an expansion of \mathcal{L}' iff \mathcal{L}' is a subset of \mathcal{L} (i.e., the sets of function symbols and predicate symbols of \mathcal{L}' are subsets of the corresponding sets of \mathcal{L}).
2. If \mathcal{L}' is a reduct of \mathcal{L} , an \mathcal{L}' -structure $\mathfrak{S}' = \langle \mathcal{D}', \mathcal{I}' \rangle$ is a reduct of an \mathcal{L} -structure $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ if $\mathcal{D}' = \mathcal{D}$ and \mathcal{I}' is the restriction of \mathcal{I} to \mathcal{L}' . \mathfrak{S} is called an expansion of \mathfrak{S}' .
3. Given a set S of signed formulae, the reduct of \mathcal{L} with respect to S , denoted \mathcal{L}_S , is the subset of \mathcal{L} consisting of all function and predicate symbols occurring in formulae in S .

Remark 2. If a language \mathcal{L}' is a reduct of a language \mathcal{L} , any \mathcal{L}' -structure \mathfrak{S}' can be expanded to an \mathcal{L} -structure \mathfrak{S} (in many ways). Furthermore, for any \mathcal{L}' -formula A and any state $\sigma^{\mathfrak{S}'}$, $\text{Val}_{\sigma^{\mathfrak{S}'}}^{\mathfrak{S}'}(A) = \mathbf{T}$ iff $\text{Val}_{\sigma^{\mathfrak{S}'}}^{\mathfrak{S}}(A) = \mathbf{T}$.

Since the language \mathcal{L} without equality is a reduct of the language \mathcal{L}_{\approx} with equality, the remark implies that all the results from this section remain valid for \mathcal{L} .

5 The Gentzen-Style System \mathbf{G}

In this section we consider the language without equality \mathcal{L} and present a Gentzen-style sequent calculus for it.

Definition 21. A sequent is a pair of sequences of formulae. A sequent $\langle \Gamma, \Delta \rangle$ is denoted by $\Gamma \rightarrow \Delta$.

In a sequent $\Gamma \rightarrow \Delta$, Γ is called the *antecedent* and Δ is called the *succedent*. If Γ is the empty sequence, the corresponding sequent is denoted as $\rightarrow \Delta$. If Δ is empty, the corresponding sequent is denoted as $\Gamma \rightarrow$. If both Γ and Δ are empty, we have the special sequent \rightarrow (the *inconsistent sequent*).

Below the symbols Γ, Δ, Λ will be used to denote arbitrary sequences of formulae² and A, B to denote formulae.

Definition 22. The sequent calculus \mathbf{G} consists of the following 16 inference rules.

$$\frac{\Gamma, A, B, \Delta \rightarrow \Lambda}{\Gamma, A \wedge B, \Delta \rightarrow \Lambda} (\rightarrow \wedge) \quad \frac{\Gamma \rightarrow \Delta, A, \Lambda \quad \Gamma \rightarrow \Delta, B, \Lambda}{\Gamma \rightarrow \Delta, A \wedge B, \Lambda} (\wedge \rightarrow)$$

² Note that Γ, Δ, Λ are sequence variables on the meta level.

$$\begin{array}{c}
\frac{\Gamma, A, \Delta \rightarrow \Lambda \quad \Gamma, B, \Delta \rightarrow \Lambda}{\Gamma, A \vee B, \Delta \rightarrow \Lambda} (\vee \rightarrow) \quad \frac{\Gamma \rightarrow \Delta, A, B, \Lambda}{\Gamma \rightarrow \Delta, A \vee B, \Lambda} (\rightarrow \vee) \\
\frac{\Gamma \rightarrow \Delta, A, \Lambda \quad \Gamma, B, \Lambda \rightarrow \Delta}{\Gamma, A \Rightarrow B, \Lambda \rightarrow \Delta} (\Rightarrow \rightarrow) \quad \frac{\Gamma, A \rightarrow \Delta, B, \Lambda}{\Gamma \rightarrow \Delta, A \Rightarrow B, \Lambda} (\rightarrow \Rightarrow) \\
\frac{\Gamma, \Delta \rightarrow A, \Lambda}{\Gamma, \neg A, \Delta \rightarrow \Lambda} (\neg \rightarrow) \quad \frac{A, \Gamma \rightarrow \Delta, \Lambda}{\Gamma \rightarrow \Delta, \neg A, \Lambda} (\rightarrow \neg)
\end{array}$$

In the quantifier rules below

1. x is any individual variable;
2. y is any individual variable free for x in A and $y \notin \mathcal{FVar}(A) \setminus \{x\}$;
3. t is any individual term free for x in A ;
4. \bar{x} is any sequence variable;
5. \bar{y} is any sequence variable free for \bar{x} in A and $\bar{y} \notin \mathcal{FVar}(A) \setminus \{\bar{x}\}$;
6. $s_1, \dots, s_n, n \geq 0$, is any sequence of terms each of them free for \bar{x} in A .

$$\begin{array}{c}
\frac{\Gamma, A\{x \mapsto t\}, \forall x A, \Delta \rightarrow \Lambda}{\Gamma, \forall x A, \Delta \rightarrow \Lambda} (\forall_I \rightarrow) \quad \frac{\Gamma \rightarrow \Delta, A\{x \mapsto y\}, \Lambda}{\Gamma \rightarrow \Delta, \forall x A, \Lambda} (\rightarrow \forall_I) \\
\frac{\Gamma, A\{x \mapsto y\}, \Delta \rightarrow \Lambda}{\Gamma, \exists x A, \Delta \rightarrow \Lambda} (\exists_I \rightarrow) \quad \frac{\Gamma \rightarrow \Delta, A\{x \mapsto t\}, \exists x A, \Lambda}{\Gamma \rightarrow \Delta, \exists x A, \Lambda} (\rightarrow \exists_I) \\
\frac{\Gamma, A\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\}, \forall \bar{x} A, \Delta \rightarrow \Lambda}{\Gamma, \forall \bar{x} A, \Delta \rightarrow \Lambda} (\forall_S \rightarrow) \quad \frac{\Gamma \rightarrow \Delta, A\{\bar{x} \mapsto \bar{y}\}, \Lambda}{\Gamma \rightarrow \Delta, \forall \bar{x} A, \Lambda} (\rightarrow \forall_S) \\
\frac{\Gamma, A\{\bar{x} \mapsto \bar{y}\}, \Delta \rightarrow \Lambda}{\Gamma, \exists \bar{x} A, \Delta \rightarrow \Lambda} (\exists_S \rightarrow) \quad \frac{\Gamma \rightarrow \Delta, A\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\}, \exists \bar{x} A, \Lambda}{\Gamma \rightarrow \Delta, \exists \bar{x} A, \Lambda} (\rightarrow \exists_S)
\end{array}$$

Note that in both the $((\rightarrow \forall_{\text{Ind}}))$ -rule and the $((\exists_{\text{Ind}} \rightarrow))$ -rule, the variable y does not occur free in the lower sequent. Similarly, in both the $((\rightarrow \forall_{\text{Seq}}))$ -rule and the $((\exists_{\text{Seq}} \rightarrow))$ -rule, the variable \bar{y} does not occur free in the lower sequent. In these rules, the variables y and \bar{y} are called the *eigenvariables* of the inference. The condition that the eigenvariable does not occur free in the conclusion of the rule is called the *eigenvariable condition*. The formulae $\forall x A$, $\exists x A$, $\forall \bar{x} A$ and $\exists \bar{x} A$ are called *principal formulae* of the inference rules $(\rightarrow \forall_{\text{Ind}})$, $(\exists_{\text{Ind}} \rightarrow)$, $(\rightarrow \forall_{\text{Seq}})$, and $(\exists_{\text{Seq}} \rightarrow)$, respectively.

In the inference rules, the upper sequents are called *premises* and the lower sequents are called *conclusions*.

The *axioms* of \mathbf{G} are all sequents $\Gamma \rightarrow \Delta$ such that Γ and Δ contain a common formula.

Definition 23.

1. A sequent $A_1, \dots, A_m \rightarrow B_1, \dots, B_n$ is falsifiable iff for some \mathcal{L} -structure \mathfrak{S} and some state $\sigma^{\mathfrak{S}}$ over \mathfrak{S}

$$\text{Val}_{\sigma}^{\mathfrak{S}}(A_1 \wedge \dots \wedge A_m \wedge \neg B_1 \wedge \dots \wedge \neg B_n) = \mathbf{T}.$$

2. A sequent $A_1, \dots, A_m \rightarrow B_1, \dots, B_n$ is valid, denoted $\models A_1, \dots, A_m \rightarrow B_1, \dots, B_n$, iff for every \mathcal{L} -structure \mathfrak{S} and for every state $\sigma^\mathfrak{S}$ over \mathfrak{S}

$$\mathcal{V}al_\sigma^\mathfrak{S}(\neg A_1 \vee \dots \vee \neg A_m \vee B_1 \vee \dots \vee B_n) = \mathbf{T}.$$

It is easy to prove the following lemma:

Lemma 5. *No axiom is falsifiable. Equivalently, every axiom is valid.*

The notion of proof is the central concept in any proof system. Proof trees are given by the following inductive definition.

Definition 24. *The set of proof trees is the least set of trees containing all one-node trees labeled with an axiom, and closed under the rules of the system \mathbf{G} in the following sense:*

1. For any proof tree T_1 whose root is labeled with a sequent $\Gamma \rightarrow \Delta$, for any instance of a one-premise inference rule with premise $\Gamma \rightarrow \Delta$ and conclusion $\Lambda \rightarrow \Psi$, the tree T whose root is labeled with $\Lambda \rightarrow \Psi$ and whose subtree $T/1$ is equal to T_1 , is a proof tree.
2. For any two proof trees T_1 and T_2 whose roots are labeled with sequents $\Gamma \rightarrow \Delta$ and $\Gamma' \rightarrow \Delta'$ respectively, for every instance of a two-premise inference rule with premises $\Gamma \rightarrow \Delta$ and $\Gamma' \rightarrow \Delta'$ and conclusion $\Lambda \rightarrow \Psi$, the tree whose root is labeled with $\Lambda \rightarrow \Psi$ and whose subtrees $T/1$ and $T/2$ are equal to T_1 and T_2 respectively, is a proof tree.

The set of deduction trees is defined inductively as the least set of trees containing all one-node trees (not necessarily labeled with an axiom), and closed under 1 and 2 as above.

The sequent labeling the root of a proof tree (deduction tree) is called the *conclusion* of the proof tree (deduction tree). A sequent is *provable* iff there exists a proof tree of which it is the conclusion. If a sequent $\Gamma \rightarrow \Delta$ is provable, this is denoted by $\vdash \Gamma \rightarrow \Delta$.

Sometimes we identify a node of a tree with the sequent that is the label of the node. A *closed leaf* of a tree is a leaf labeled with an axiom. A *closed tree* is a tree with all its leaves closed.

6 Soundness of \mathbf{G}

In order to establish soundness of the system \mathbf{G} , the following lemmata are needed.

Lemma 6. *Let \mathfrak{S} be an \mathcal{L} -structure and $\sigma^\mathfrak{S}$ be a state over \mathfrak{S} .*

1. For any term r and any individual term t ,

$$\mathcal{V}al_\sigma^\mathfrak{S}(r\{x \mapsto t\}) = \mathcal{V}al_\sigma^\mathfrak{S}(r\{x \mapsto d_\mathfrak{S}\}),$$

where $\mathcal{V}al_\sigma^\mathfrak{S}(t) = d$.

2. For any terms $r, t_1, \dots, t_n, n \geq 0$,

$$\mathcal{V}al_{\sigma}^{\mathfrak{S}}(r\{\bar{x} \mapsto \ulcorner t_1, \dots, t_n \urcorner\}) = \mathcal{V}al_{\sigma}^{\mathfrak{S}}(r\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{m\mathfrak{S}} \urcorner\}),$$

where $\ulcorner \mathcal{V}al_{\sigma}^{\mathfrak{S}}(t_1), \dots, \mathcal{V}al_{\sigma}^{\mathfrak{S}}(t_n) \urcorner = \ulcorner d_1, \dots, d_m \urcorner$.

3. If t is an individual term free for x in a formula F . Then

$$\mathcal{V}al_{\sigma}^{\mathfrak{S}}(F\{x \mapsto t\}) = \mathcal{V}al_{\sigma}^{\mathfrak{S}}(F\{x \mapsto d_{\mathfrak{S}}\}),$$

where $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(t) = d$.

4. Let $t_1, \dots, t_n, n \geq 0$, be terms free for \bar{x} in a formula F . Then

$$\mathcal{V}al_{\sigma}^{\mathfrak{S}}(F\{\bar{x} \mapsto \ulcorner t_1, \dots, t_n \urcorner\}) = \mathcal{V}al_{\sigma}^{\mathfrak{S}}(F\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{m\mathfrak{S}} \urcorner\}),$$

where $\ulcorner \mathcal{V}al_{\sigma}^{\mathfrak{S}}(t_1), \dots, \mathcal{V}al_{\sigma}^{\mathfrak{S}}(t_n) \urcorner = \ulcorner d_1, \dots, d_m \urcorner$.

Proof. The lemma can be proved using the induction on terms and formulae. \square

Lemma 7. For any rule of the calculus \mathbf{G} , the conclusion is falsifiable in some \mathcal{L} -structure \mathfrak{S} if and only if one of the premises is falsifiable in \mathfrak{S} . Equivalently, the conclusion is valid if and only if all premises are valid.

Proof. We prove the lemma for the rules $(\rightarrow \forall_{\text{Seq}})$ and $(\forall_{\text{Seq}} \rightarrow)$. We use the following abbreviations: If Γ is the antecedent of a sequent, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\Gamma) = \mathbf{T}$ means that $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(F) = \mathbf{T}$ for every formula $F \in \Gamma$, and if Δ is the succedent of a sequent, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\Delta) = \mathbf{F}$ means that $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(F) = \mathbf{F}$ for every formula $F \in \Delta$.

$(\rightarrow \forall_{\text{Seq}})$. Assume that the sequent $\Gamma \rightarrow \Delta, A\{\bar{x} \mapsto \bar{y}\}, \Lambda$ is falsifiable. This means that there is a \mathcal{L} -structure $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ and a state $\sigma^{\mathfrak{S}}$ over \mathfrak{S} such that $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\Gamma) = \mathbf{T}$, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\Delta) = \mathbf{F}$, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \bar{y}\}) = \mathbf{F}$, and $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\Lambda) = \mathbf{F}$.

Recall that $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\forall \bar{x} A) = \mathbf{T}$ iff $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \{d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}}\}\}) = \mathbf{T}$ for all $d_1, \dots, d_n \in \mathcal{D}$.

Since \bar{y} is free for \bar{x} in A , by Lemma 6, for $\ulcorner d_1, \dots, d_m \urcorner = \mathcal{V}al_{\sigma}^{\mathfrak{S}}(\bar{y})$ we have $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \bar{y}\}) = \mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{m\mathfrak{S}} \urcorner\})$. Hence, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\forall \bar{x} A) = \mathbf{F}$, which implies that $\Gamma \rightarrow \Delta, \forall \bar{x} A, \Lambda$ is falsified by \mathfrak{S} and $\sigma^{\mathfrak{S}}$.

Conversely, assume that $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ and $\sigma^{\mathfrak{S}}$ falsify $\Gamma \rightarrow \Delta, \forall \bar{x} A, \Lambda$. In particular, there are some $d_1, \dots, d_n \in \mathcal{D}$ such that $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) = \mathbf{F}$. Since $\bar{y} \in \mathcal{F}Var(A) \setminus \{\bar{x}\}$ and \bar{y} is free for \bar{x} in A , by Lemma 3 and Lemma 7, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}[\bar{y} := \ulcorner d_1, \dots, d_n \urcorner](A\{\bar{x} \mapsto \bar{y}\}) = \mathbf{F}$. Since \bar{y} does not appear free in Γ, Δ or Λ , by Lemma 3 we also have $\mathcal{V}al_{\sigma}^{\mathfrak{S}}[\bar{y} := \ulcorner d_1, \dots, d_n \urcorner](\Gamma) = \mathbf{T}$, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}[\bar{y} := \ulcorner d_1, \dots, d_n \urcorner](\Delta) = \mathbf{F}$, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}[\bar{y} := \ulcorner d_1, \dots, d_n \urcorner](\Lambda) = \mathbf{F}$. Hence, \mathfrak{S} and $\sigma^{\mathfrak{S}}[\bar{y} := \ulcorner d_1, \dots, d_n \urcorner]$ falsify the sequent $\Gamma \rightarrow \Delta, A\{\bar{x} \mapsto \bar{y}\}, \Lambda$.

$(\forall_{\text{Seq}} \rightarrow)$. Assume \mathfrak{S} and $\sigma^{\mathfrak{S}}$ falsify $\Gamma, A\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\}, \forall \bar{x} A, \Delta \rightarrow \Lambda$. Then it is clear that \mathfrak{S} and $\sigma^{\mathfrak{S}}$ falsify also $\Gamma, \forall \bar{x} A, \Delta \rightarrow \Lambda$.

Conversely, assume that $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ and $\sigma^{\mathfrak{S}}$ falsify $\Gamma, \forall \bar{x} A, \Delta \rightarrow \Lambda$. Then $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\Gamma) = \mathbf{T}$, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\forall \bar{x} A) = \mathbf{T}$, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\Delta) = \mathbf{T}$, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(\Lambda) = \mathbf{F}$. In particular, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{m\mathfrak{S}} \urcorner\}) = \mathbf{T}$ for every $d_1, \dots, d_m \in \mathcal{D}$.

By Lemma 6, for $\ulcorner \mathcal{V}al_{\sigma}^{\mathfrak{S}}(s_1), \dots, \mathcal{V}al_{\sigma}^{\mathfrak{S}}(s_n) \urcorner = \ulcorner d_1, \dots, d_m \urcorner$ we have that $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\}) = \mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{m\mathfrak{S}} \urcorner\})$ and, therefore, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(A\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\}) = \mathbf{T}$. Hence, \mathfrak{S} and $\sigma^{\mathfrak{S}}$ falsify also the sequent $\Gamma, A\{\bar{x} \mapsto \ulcorner s_1, \dots, s_n \urcorner\}, \forall \bar{x} A, \Delta \rightarrow \Lambda$. \square

These lemmata imply soundness of \mathbf{G} :

Theorem 1 (Soundness of \mathbf{G}). *Every sequent provable in \mathbf{G} is valid.*

7 Completeness of \mathbf{G}

7.1 Hintikka Sets over Languages without Equality

Now we define Hintikka sets over languages without equality.

Definition 25. *A Hintikka set S over a language \mathcal{L} without equality with respect to a term algebra \mathcal{H} (over the reduct \mathcal{L}_S) is a set of signed \mathcal{L} -formulae such that the following conditions hold for all signed formulae A, B, C, D of type α, β, γ and δ :*

- H0 No atomic formula and its conjugate are both in S ($\top A$ or FA is atomic iff A is).*
- H1 If a type- α formula A is in S , then both A_1 and A_2 are in S .*
- H2 If a type- β formula B is in S , then either B_1 is in S or B_2 is in S .*
- H3 If a type- γ_i formula C is in S , then for every $t \in \mathcal{H}_{\text{Ind}}$, $C(t)$ is in S (we require that t is free for x in C for every $t \in \mathcal{H}_{\text{Ind}}$).*
- H4 If a type- γ_s formula C is in S , then for every $n \geq 0$ and terms $s_1, \dots, s_n \in \mathcal{H}$, $C(s_1, \dots, s_n)$ is in S (we require that s_1, \dots, s_n are free for \bar{x} in C for every $s_1, \dots, s_n \in \mathcal{H}$).*
- H5 If a type- δ_i formula D is in S , then there exists $t \in \mathcal{H}_{\text{Ind}}$ such that $D(t)$ is in S (we require that t is free for x in D for every $t \in \mathcal{H}_{\text{Ind}}$).*
- H6 If a type- δ_s formula D is in S , then there exist terms $s_1, \dots, s_n \in \mathcal{H}$ such that $D(s_1, \dots, s_n)$ is in S (we require that s_1, \dots, s_n are free for \bar{x} in D for every $s_1, \dots, s_n \in \mathcal{H}$).*
- H7 Every (individual or sequence) variable v occurring free in some formula of S is in \mathcal{H} .*

The condition *H7* and the fact that \mathcal{H} is a term algebra imply that, for every term occurring in some formula in S , if the term is closed or contains only variables free in S , then it is in \mathcal{H} .

Lemma 8. *Every Hintikka set S (with respect to a term algebra \mathcal{H}) is satisfiable in a structure \mathfrak{H}_S with the domain \mathcal{H} .*

Proof. The \mathcal{L}_S -structure $\mathfrak{H}_S = \langle \mathcal{H}, \mathcal{I} \rangle$ is defined as follows:

Every individual or sequence constant c in \mathcal{L}_S is interpreted as the term c .

Every n -ary individual or sequence function symbol f in \mathcal{L}_S is interpreted such that, for any terms $t_1, \dots, t_n \in \mathcal{H}_{\text{Ind}}$, $f_{\mathcal{I}}(t_1, \dots, t_n) = f(t_1, \dots, t_n)$.

Every flexible arity individual or sequence function symbol f in \mathcal{L}_S is interpreted such that, for any $n \geq 0$ and terms $s_1, \dots, s_n \in \mathcal{H}$, $f_{\mathcal{I}}(s_1, \dots, s_n) = f(s_1, \dots, s_n)$.

For every n -ary predicate symbol p in \mathcal{L}_S and for any terms $t_1, \dots, t_n \in \mathcal{H}_{\text{Ind}}$

$$p_{\mathcal{I}}(t_1, \dots, t_n) = \begin{cases} \mathbf{T}, & \text{if } \top p(t_1, \dots, t_n) \in S \\ \mathbf{F}, & \text{if } \text{Fp}(t_1, \dots, t_n) \in S \\ & \text{or neither } \top p(t_1, \dots, t_n) \text{ nor } \text{Fp}(t_1, \dots, t_n) \text{ is in } S. \end{cases}$$

For every flexible arity predicate symbol p in \mathcal{L}_S , for any $n \geq 0$ and for terms $s_1, \dots, s_n \in \mathcal{H}$

$$p_{\mathcal{I}}(s_1, \dots, s_n) = \begin{cases} \mathbf{T}, & \text{if } \top p(s_1, \dots, s_n) \in S \\ \mathbf{F}, & \text{if } \text{Fp}(s_1, \dots, s_n) \in S \\ & \text{or neither } \top p(s_1, \dots, s_n) \text{ nor } \text{Fp}(s_1, \dots, s_n) \text{ is in } S. \end{cases}$$

Let $\sigma^{\mathfrak{H}_S}$ be an assignment that is identity on the variables belonging to \mathcal{H} . First we show that for every term $t \in \mathcal{H}$, $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(t) = t$.

Let t be an individual or sequence variable v . Then $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(v) = \sigma^{\mathfrak{H}_S}(v) = v$.

Let t be an individual or sequence constant c . Then $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(c) = c_{\mathfrak{H}_S} = c$.

Let t be $f(t_1, \dots, t_n)$, where f is an individual or sequence function symbol (of fixed or flexible arity). Then we have the value $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(f(t_1, \dots, t_n)) = f_{\mathcal{I}}(\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(t_1), \dots, \mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(t_n)) = f_{\mathcal{I}}(t_1, \dots, t_n) = f(t_1, \dots, t_n)$.

Now we show by induction on formulae that for any signed formula $A \in S$, $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(A) = \mathbf{T}$.

Let A be $\top p(t_1, \dots, t_n)$, where p is a fixed or flexible arity predicate symbol. By *H7*, the variables in t_1, \dots, t_n are in \mathcal{H} , and since \mathcal{H} is a terms algebra, t_1, \dots, t_n are in \mathcal{H} . Then by definition of $p_{\mathcal{I}}$ and the fact that $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(t) = t$ for any term $t \in \mathcal{H}$ we get $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(p(t_1, \dots, t_n)) = p_{\mathcal{I}}(\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(t_1), \dots, \mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(t_n)) = p_{\mathcal{I}}(t_1, \dots, t_n) = \mathbf{T}$, which implies $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(A) = \mathbf{T}$.

In the similar way it can be shown that if A is $\text{Fp}(t_1, \dots, t_n) \in S$, then $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(p(t_1, \dots, t_n)) = \mathbf{F}$ and, hence, $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(A) = \mathbf{T}$.

Proof for type- α and type- β formulae is easy. We prove now for type- γ_s and type- δ_s formulae.

If a signed formula C of type γ_s is in S , $C(s_1, \dots, s_n)$ is in S for any terms s_1, \dots, s_n in \mathcal{H} and any $n \geq 0$ by *H4*. Since $C(s_1, \dots, s_n)$ contains one less quantifier than C , by the induction hypothesis $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(C(s_1, \dots, s_n)) = \mathbf{T}$ for every s_1, \dots, s_n in \mathcal{H} and for every $n \geq 0$. By Lemma 6 and the fact that $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(t) = t$, we get $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(C(s_1, \dots, s_n)) = \mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(C(s_{1_{\mathfrak{H}_S}}, \dots, s_{n_{\mathfrak{H}_S}})) = \mathbf{T}$ for any $n \geq 0$ and terms s_1, \dots, s_n in \mathcal{H} . By Lemma 4, it implies $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(C) = \mathbf{T}$.

If a signed formula D of type δ_s is in S , $D(s_1, \dots, s_n)$ is in S for some s_1, \dots, s_n in \mathcal{H} by *H6*. Since $D(s_1, \dots, s_n)$ contains one less quantifier than D , by the induction hypothesis $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(D(s_1, \dots, s_n)) = \mathbf{T}$ for some s_1, \dots, s_n in \mathcal{H} . By Lemma 6 and the fact that $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(t) = t$, we get $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(D(s_1, \dots, s_n)) = \mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(D(s_{1_{\mathfrak{H}_S}}, \dots, s_{n_{\mathfrak{H}_S}})) = \mathbf{T}$ for some terms s_1, \dots, s_n in \mathcal{H} . By Lemma 4, it implies $\mathcal{V}al_{\sigma}^{\mathfrak{H}_S}(D) = \mathbf{T}$. Finally, using Remark 2, \mathfrak{H}_S can be expanded to an \mathcal{L} -structure satisfying S . \square

The domain \mathcal{H} of \mathfrak{H}_S is called a *Herbrand universe*.

Now, given a (possibly infinite) sequent $\Gamma \rightarrow \Delta$, our goal is to attempt to falsify it. For this, we design the **Search** procedure with the following properties:

1. If the original sequent is valid, the **Search** procedure stops after a finite number of steps, yielding a proof tree.
2. If the original sequent is falsifiable, the **Search** procedure constructs a possibly infinite tree, and along some (possibly infinite) path in the tree it can be shown that a Hintikka set exists, which yields a counter example for the sequent.

Given a sequent $\Gamma \rightarrow \Delta$, we assume that the set of all variables occurring free in some formula in the sequent is disjoint from the set of all variables bound in some formula in the sequent. It ensures that terms occurring in formulae in the sequent are free for substitutions. Assume for convenience also that distinct quantifiers bind distinct variables.

For proving correctness of the **Search** procedure it is convenient to have a slightly more general version of the quantifier rules $(\forall_{\text{Ind}} \rightarrow)$, $(\rightarrow \exists_{\text{Ind}})$, $(\forall_{\text{Seq}} \rightarrow)$ and $(\rightarrow \exists_{\text{Seq}})$.

Definition 26. *The extended rules $(\forall_{\text{Ind}} \rightarrow)$, $(\rightarrow \exists_{\text{Ind}})$, $(\forall_{\text{Seq}} \rightarrow)$ and $(\rightarrow \exists_{\text{Seq}})$ are the following:*

$$\frac{\Gamma, A\{x \mapsto t_1\}, \dots, A\{x \mapsto t_k\}, \forall x A, \Delta \rightarrow \Lambda}{\Gamma, \forall x A, \Delta \rightarrow \Lambda} (\forall_{\text{Ind}} \rightarrow)$$

$$\frac{\Gamma \rightarrow \Delta, A\{x \mapsto t_1\}, \dots, A\{x \mapsto t_k\}, \exists x A, \Lambda}{\Gamma \rightarrow \Delta, \exists x A, \Lambda} (\rightarrow \exists_{\text{Ind}})$$

where t_1, \dots, t_k are any k terms ($k \geq 1$) free for x in A .

$$\frac{\Gamma, A\{\bar{x} \mapsto \ulcorner s_1^1, \dots, s_{n_1}^1 \urcorner\}, \dots, A\{\bar{x} \mapsto \ulcorner s_1^k, \dots, s_{n_k}^k \urcorner\}, \forall \bar{x} A, \Delta \rightarrow \Lambda}{\Gamma, \forall \bar{x} A, \Delta \rightarrow \Lambda} (\forall_{\text{Seq}} \rightarrow)$$

$$\frac{\Gamma \rightarrow \Delta, A\{\bar{x} \mapsto \ulcorner s_1^1, \dots, s_{n_1}^1 \urcorner\}, \dots, A\{\bar{x} \mapsto \ulcorner s_1^k, \dots, s_{n_k}^k \urcorner\}, \exists \bar{x} A, \Lambda}{\Gamma \rightarrow \Delta, \exists \bar{x} A, \Lambda} (\rightarrow \exists_{\text{Seq}})$$

where $\ulcorner s_1^1, \dots, s_{n_1}^1 \urcorner, \dots, \ulcorner s_1^k, \dots, s_{n_k}^k \urcorner$ are any k sequences ($k \geq 1$) of terms free for \bar{x} in A .

In fact, they can be simulated by k applications of the corresponding old rules from **G**, but the advantage of using the new rules is that they may reduce the size of proof trees. From now on we assume that the rules of Definition 26 are used as the $(\forall_{\text{Ind}} \rightarrow)$, $(\rightarrow \exists_{\text{Ind}})$, $(\forall_{\text{Seq}} \rightarrow)$ and $(\rightarrow \exists_{\text{Seq}})$ rules of the Gentzen system **G**.

7.2 Organizing the Terms

We assume without loss of generality that the set \mathcal{V} is the disjoint union of the countably infinite sets $\mathcal{V}^{\text{Used}} = \{u_0, u_1, u_2, \dots\}$ and $\mathcal{V}^{\text{New}} = \{v_0, v_1, v_2, \dots\}$ and that only variables in $\mathcal{V}^{\text{Used}}$ are used to build formulae. In this way, \mathcal{V}^{New} is an infinite supply of new variables. We denote by $\mathcal{V}_{\text{Ind}}^{\text{Used}}$ the set $\mathcal{V}^{\text{Used}} \cap \mathcal{V}_{\text{Ind}}$, by

$\mathcal{V}_{\text{Seq}}^{\text{Used}}$ the set $\mathcal{V}^{\text{Used}} \cap \mathcal{V}_{\text{Seq}}$, by $\mathcal{V}_{\text{Ind}}^{\text{New}}$ the set $\mathcal{V}^{\text{New}} \cap \mathcal{V}_{\text{Ind}}$, and by $\mathcal{V}_{\text{Seq}}^{\text{New}}$ the set $\mathcal{V}^{\text{New}} \cap \mathcal{V}_{\text{Seq}}$.

Let \mathcal{L}' be a reduct of \mathcal{L} consisting of the constant, function and predicate symbols occurring in formulae in $\Gamma \rightarrow \Delta$.

Let $\mathcal{CV}_{\text{Ind}}$ be the set of all individual constants and free individual variables in $\Gamma \rightarrow \Delta$, and $\mathcal{CV}_{\text{Seq}}$ be the set of all sequence constants and free sequence variables in $\Gamma \rightarrow \Delta$. Then we define the set $\mathcal{T}erms$ as follows:

1. If $\mathcal{CV}_{\text{Ind}} \neq \emptyset$ and $\mathcal{CV}_{\text{Seq}} \neq \emptyset$, then $\mathcal{T}erms = \{t_1, t_2, \dots\}$ is a set of all \mathcal{L}' -terms constructed from the variables occurring free in formulae in $\Gamma \rightarrow \Delta$, and the constant and function symbols in \mathcal{L}' .
2. If $\mathcal{CV}_{\text{Ind}} \neq \emptyset$ and $\mathcal{CV}_{\text{Seq}} = \emptyset$, then $\mathcal{T}erms = \{\overline{y_0}, t_1, t_2, \dots\}$ is a set of all \mathcal{L}' -terms constructed from the sequence variable $\overline{y_0} \in \mathcal{V}_{\text{Seq}}^{\text{New}}$, the individual variables occurring free in formulae in $\Gamma \rightarrow \Delta$, the individual constants in $\Gamma \rightarrow \Delta$ and the function symbols in \mathcal{L}' .
3. If $\mathcal{CV}_{\text{Ind}} = \emptyset$ and $\mathcal{CV}_{\text{Seq}} \neq \emptyset$, then $\mathcal{T}erms = \{y_0, t_1, t_2, \dots\}$ is a set of all \mathcal{L}' -terms constructed from the individual variable $y_0 \in \mathcal{V}_{\text{Ind}}^{\text{New}}$, the sequence variables occurring free in formulae in $\Gamma \rightarrow \Delta$, the sequence constants in $\Gamma \rightarrow \Delta$ and the function symbols in \mathcal{L}' .
4. If $\mathcal{CV}_{\text{Ind}} = \emptyset$ and $\mathcal{CV}_{\text{Seq}} = \emptyset$, then $\mathcal{T}erms = \{y_0, \overline{y_0}, t_1, t_2, \dots\}$ is a set of all \mathcal{L}' -terms constructed from the individual variable $y_0 \in \mathcal{V}_{\text{Ind}}^{\text{New}}$, from the sequence variable $\overline{y_0} \in \mathcal{V}_{\text{Seq}}^{\text{New}}$, and the function symbols in \mathcal{L}' .

We denote by $\mathcal{T}uples$ an enumeration of the set of all finite tuples constructed from the elements of $\mathcal{T}erms$. Note that the empty tuple $\langle \rangle$ is in $\mathcal{T}uples$. The list $\mathcal{I}nds$ is an enumeration of the subset of $\mathcal{T}erms$ consisting of all individual terms in $\mathcal{T}erms$.

Let

$$\mathcal{T}up_0 = [[\text{First}(\mathcal{T}uples), []]],$$

where the function **First** selects the first element from the list it is applied to. Let also

$$\mathcal{T}up_0^{\text{Av}} = \text{Tail}(\mathcal{T}uples),$$

where the function **Tail** returns the tail of the list it is applied to. Moreover, we define

$$\mathcal{I}nd_0 = [[\text{First}(\mathcal{I}nds), []]]$$

and

$$\mathcal{I}nd_0^{\text{Av}} = \text{Tail}(\mathcal{I}nds).$$

In general, during an attempt to prove $\Gamma \rightarrow \Delta$ we keep track of which individual terms and which tuples of terms have been thus far activated. The list of activated individual terms is kept in $\mathcal{I}nd_0$ and the list of activated tuples of terms is kept in $\mathcal{T}up_0$.

In order to handle the rules $(\forall_{\text{Ind}} \rightarrow)$ or $(\rightarrow \exists_{\text{Ind}})$ correctly, we structure $\mathcal{I}nd_0$ as a list of lists, where each list $[t_i, \mathcal{I}\mathcal{F}ml_0(i)]$ consists of an individual term t_i and a list $\mathcal{I}\mathcal{F}ml_0(i)$ of the formulae $\forall xA$ (or $\exists xA$), such that t_i was

used as a term t for the rule $(\forall_{\text{Ind}} \rightarrow)$ (or $(\rightarrow \exists_{\text{Ind}})$) with the principal formula $\forall xA$ (or $\exists xA$). Initially, each $\mathcal{I}Fml_0(i)$ is the empty list. The lists $\mathcal{I}Fml_0(i)$ are updated each time when the term t_i is used in a rule $(\forall_{\text{Ind}} \rightarrow)$ or $(\rightarrow \exists_{\text{Ind}})$. We also let $it(\mathcal{I}nd_0)$ denote the set of individual terms $\{t_i \mid [t_i, \mathcal{I}Fml_0(i)] \in \mathcal{I}nd_0\}$.

Similarly, in order to handle the rules $(\forall_{\text{Seq}} \rightarrow)$ or $(\rightarrow \exists_{\text{Seq}})$ correctly, we structure $\mathcal{T}up_0$ as a list of lists, where each list $[(t_1^i, \dots, t_n^i), \mathcal{S}Fml_0(i)]$ consists of a tuple of terms $\langle t_1^i, \dots, t_n^i \rangle$ and a list $\mathcal{S}Fml_0(i)$ of the formulae $\forall \bar{x}A$ (or $\exists \bar{x}A$), such that the sequence $\ulcorner t_1^i, \dots, t_n^i \urcorner$ was used as a sequence $\ulcorner s_1, \dots, s_n \urcorner$ for the rule $(\forall_{\text{Seq}} \rightarrow)$ (or $(\rightarrow \exists_{\text{Seq}})$) with the principal formula $\forall \bar{x}A$ (or $\exists \bar{x}A$). Initially, each $\mathcal{S}Fml_0(i)$ is the empty list. The lists $\mathcal{S}Fml_0(i)$ are updated each time when the sequence $\ulcorner t_1^i, \dots, t_n^i \urcorner$ is used in a rule $(\forall_{\text{Seq}} \rightarrow)$ or $(\rightarrow \exists_{\text{Seq}})$. We also let $tup(\mathcal{T}up_0)$ denote the set of tuples of terms $\{\langle t_1^i, \dots, t_n^i \rangle \mid [t_1^i, \dots, t_n^i], \mathcal{S}Fml_0(i)] \in \mathcal{T}up_0\}$.

For any $i \geq 1$, let $\mathcal{A}vail_i$ be a set of all \mathcal{L}' -terms actually containing some occurrence of $y_i \in \mathcal{V}_{\text{Ind}}^{\text{New}}$, some occurrence of $\bar{y}_i \in \mathcal{V}_{\text{Seq}}^{\text{New}}$, and constructed from the variables occurring free in the formulae of $\Gamma \rightarrow \Delta$, the constant and function symbols occurring in $\Gamma \rightarrow \Delta$, and the variables $y_1, \dots, y_i \in \mathcal{V}_{\text{Ind}}^{\text{New}}$ and $\bar{y}_1, \dots, \bar{y}_i \in \mathcal{V}_{\text{Seq}}^{\text{New}}$.

For any $i \geq 1$, by $\mathcal{T}up_i^{\text{Av}}$ we denote the enumeration of the set of all non-empty finite tuples constructed from the elements of $\mathcal{A}vail_i$. We assume that the first element in $\mathcal{T}up_i^{\text{Av}}$ is $\langle \bar{y}_i \rangle$. By $\mathcal{I}nd_i^{\text{Av}}$ we denote the enumeration of the subset of $\mathcal{A}vail_i$ consisting of all individual terms of $\mathcal{A}vail_i$. We assume that the first element in $\mathcal{I}nd_i^{\text{Av}}$ is y_i .

Each time a rule $(\rightarrow \forall_{\text{Ind}})$ or $(\exists_{\text{Ind}} \rightarrow)$ is applied, we use as the variable y the first individual variable y_i from $\mathcal{V}_{\text{Ind}}^{\text{New}}$, append $[y_i, \square]$ to $\mathcal{I}nd_0$ and delete y_i from the list $\mathcal{I}nd_i^{\text{Av}}$. We use a counter $\mathcal{A}ct_{\text{Ind}}$ to record the number of individual variables y_1, y_2, \dots thus far activated. Initially $\mathcal{A}ct_{\text{Ind}} = 0$, and every time a new y_i gets activated $\mathcal{A}ct_{\text{Ind}}$ is incremented by 1.

Each time a rule $(\rightarrow \forall_{\text{Seq}})$ or $(\exists_{\text{Seq}} \rightarrow)$ is applied, we use as the variable \bar{y} the first sequence variable \bar{y}_i from $\mathcal{V}_{\text{Seq}}^{\text{New}}$, append $[\langle \bar{y}_i \rangle, \square]$ to $\mathcal{T}up_0$ and delete $\langle \bar{y}_i \rangle$ from the list $\mathcal{T}up_i^{\text{Av}}$. We use a counter $\mathcal{A}ct_{\text{Tup}}$ to record the number of tuples $\langle \bar{y}_1 \rangle, \langle \bar{y}_2 \rangle, \dots$ thus far activated. Initially $\mathcal{A}ct_{\text{Tup}} = 0$, and every time a new tuple $\langle \bar{y}_i \rangle$ gets activated $\mathcal{A}ct_{\text{Tup}}$ is incremented by 1.

Each time a rule $(\forall_{\text{Ind}} \rightarrow)$ (or $(\rightarrow \exists_{\text{Ind}})$) is applied with the principal formula $\forall xA$ (or $\exists xA$), all the terms $t_k \in it(\mathcal{I}nd_0)$ such that $A \notin \mathcal{I}Fml_0(k)$, are available as terms t . Furthermore, at the end of every round, the list $[t_i, \square]$ is appended to $it(\mathcal{I}nd_0)$, where $t_i = \text{First}(\mathcal{I}nd_i^{\text{Av}})$ for all available lists $\mathcal{I}nd_i^{\text{Av}}$ with $0 \leq i \leq \mathcal{A}ct_{\text{Ind}}$. At the same time t_i is removed from $\mathcal{I}nd_i^{\text{Av}}$.

Similarly, each time a rule $(\forall_{\text{Seq}} \rightarrow)$ (or $(\rightarrow \exists_{\text{Seq}})$) is applied with the principal formula $\forall \bar{x}A$ (or $\exists \bar{x}A$), the sequences $\ulcorner t_1^k, \dots, t_n^k \urcorner$ from all the tuples $\langle t_1^k, \dots, t_n^k \rangle \in tup(\mathcal{T}up_0)$ such that $A \notin \mathcal{S}Fml_0(k)$, are available as sequences $\ulcorner s_1, \dots, s_n \urcorner$. Furthermore, at the end of every round, the list $[\langle t_1^i, \dots, t_{n_i}^i \rangle, \square]$ is appended to $tup(\mathcal{T}up_0)$, where $\langle t_1^i, \dots, t_{n_i}^i \rangle = \text{First}(\mathcal{T}up_i^{\text{Av}})$ for all available lists $\mathcal{T}up_i^{\text{Av}}$ with $0 \leq i \leq \mathcal{A}ct_{\text{Tup}}$. At the same time $\langle t_1^i, \dots, t_{n_i}^i \rangle$ is removed from $\mathcal{T}up_i^{\text{Av}}$.

7.3 The Search Procedure

The Search procedure is given on Fig. 1.

```

Procedure Search( $\Gamma_0 \rightarrow \Delta_0$  : sequent; var  $T$  : tree)
1:  $L := \text{Tail}(\Gamma_0)$ ;  $\Gamma := \text{Head}(\Gamma_0)$ 
2:  $R := \text{Tail}(\Delta_0)$ ;  $\Delta := \text{Head}(\Delta_0)$ 
3:  $T :=$  one-node tree labeled with  $\Gamma \rightarrow \Delta$ 
4: initialize  $\text{Up}_0, \text{Up}_i^{\text{Av}}, i \geq 0$ , as above
5: initialize  $\text{Ind}_0, \text{Ind}_i^{\text{Av}}, i \geq 0$ , as above
6:  $\text{Act}_{\text{Up}} := 0$ ;  $\text{Act}_{\text{Ind}} := 0$ 
7: while not all leaves of  $T$  are finished do
8:    $\text{Up}_1 := \text{Up}_0$ ;  $\text{Ind}_1 := \text{Ind}_0$ 
9:    $T_0 := T$ 
10:  for each leaf node of  $T_0$ 
11:    (in lexicographic order) do
12:      if not finished(node) then
13:        Expand(node,  $T$ )
14:      end
15:    end
16:   $\text{Up}_0 := \text{Up}_1$ ;  $\text{Ind}_0 := \text{Ind}_1$ 
17:   $L := \text{Tail}(L)$ ;  $R := \text{Tail}(R)$ 
18:  for  $i := 0$  to  $\text{Act}_{\text{Up}}$  do
19:     $F := [\text{First}(\text{Up}_i^{\text{Av}}), []]$ 
20:     $\text{Up}_0 := \text{Append}(\text{Up}_0, F)$ 
21:     $\text{Up}_i^{\text{Av}} := \text{Tail}(\text{Up}_i^{\text{Av}})$ 
22:  end
23:  for  $i := 0$  to  $\text{Act}_{\text{Ind}}$  do
24:     $F := [\text{First}(\text{Ind}_i^{\text{Av}}), []]$ 
25:     $\text{Ind}_0 := \text{Append}(\text{Ind}_0, F)$ 
26:     $\text{Ind}_i^{\text{Av}} := \text{Tail}(\text{Ind}_i^{\text{Av}})$ 
27:  end
28: end
29: if all leaves are closed then
30:   write(" $T$  is a proof of  $\Gamma_0 \rightarrow \Delta_0$ ")
31: else write(" $\Gamma_0 \rightarrow \Delta_0$  is falsifiable")
32: end

```

Fig. 1. Procedure Search.

The notion of *finished leaf*, which occurs on line 7 and 12 of the procedure Search, is defined as follows: A leaf of the tree is finished if either

1. It is an axiom, or
2. L and R are empty, and the sequent contains only atoms, or formulae $\forall xA$ (or $\exists xA$) belonging to all lists $\mathcal{IF}ml_0(i)$ for all $[t_i, \mathcal{IF}ml_0(i)]$ in Ind_0 , or formu-

lae $\forall \bar{x}A$ (or $\exists \bar{x}A$) belonging to all lists $\mathcal{SFl}_0(i)$ for all $[\langle t_1^i, \dots, t_{n_i}^k \rangle, \mathcal{SFl}_0(i)]$ in $\mathcal{T}up_0$.

The procedure **Expand** on line 13 of the **Search** procedure is given on Fig. 2. The

Procedure Expand(*node* : tree-address; **var** *T* : tree)

```

1: let  $A_1, \dots, A_m \rightarrow B_1, \dots, B_n$  be the label of node
2:  $S :=$  one-node tree
   labeled with  $A_1, \dots, A_m \rightarrow B_1, \dots, B_n$ 
3: for  $i := 1$  to  $m$  do
4:   if non-atomic( $A_i$ ) then
5:     Grow-Left( $A_i, S$ )
6:   end
7: end
8: for  $i := 1$  to  $n$  do
9:   if non-atomic( $B_i$ ) then
10:    Grow-Right( $B_i, S$ )
11:   end
12: end
13: for each leaf  $l$  in  $S$  do
14:   let  $\Gamma \rightarrow \Delta$  be a label of  $l$ 
15:    $\Gamma' := \Gamma, \text{First}(L)$ 
16:    $\Delta' := \Delta, \text{First}(R)$ 
17:   create a new node  $l_1$  labeled with  $\Gamma' \rightarrow \Delta'$ 
18:  $T := \text{substitute}(T, \text{node}, S)$ 

```

Fig. 2. Procedure **Expand**.

procedure **Grow-Left** on line 5 in the procedure **Expand** is described on Fig. 3. The procedure **Grow-Right** on line 10 in the procedure **Expand** is described on Fig. 4. The predicate *non-atomic* on lines 4 and 9 in **Expand** succeeds iff its argument is not an atomic formula. The function *substitute* on line 18 in **Expand** yields a tree $T[\text{node} \leftarrow S]$ obtained by substituting the tree S at the address *node* in the tree T . Since a sequent $A_1, \dots, A_m \rightarrow B_1, \dots, B_n$ is processed from left to right, if the propositions A_1, \dots, A_{i-1} have been expanded so far, since the propositions $A_i, \dots, A_m, B_1, \dots, B_n$ are copied unchanged, every leaf of the tree S obtained so far is of the form $\Gamma, A_i, \dots, A_m \rightarrow \Delta, B_1, \dots, B_n$. Similarly, if the propositions $A_1, \dots, A_m, B_1, \dots, B_{i-1}$ have been expanded so far, since the propositions B_i, \dots, B_n are copied unchanged, every leaf of the tree S obtained so far is of the form $\Gamma \rightarrow \Delta, B_i, \dots, B_n$.

Now we prove two lemmata that imply the completeness theorem.

Lemma 9. *If the input sequent $\Gamma_0 \rightarrow \Delta_0$ is falsifiable, then either*

1. *Procedure **Search** halts with a finite counter-example tree (a tree with a finished non-closed leaf) T and, if we let*

$$H = \text{it}(\mathcal{I}nd_0)$$

Procedure Grow-Left (A : formula; var S : tree)	
1: case A of	
2: $B \wedge C, B \vee C$	$B \Rightarrow C, \neg B$: expand every non-axiom leaf of S using the left rule corresponding to the main propositional connectives;
3: $\forall xB$: for every term $t_k \in it(\mathcal{I}nd_0)$ such that $A \notin \mathcal{I}Fml_0(k)$ do extend every non-axiom leaf of S by applying the $(\forall_{\text{Ind}} \rightarrow)$ rule using the term t_k as one of the terms of the rule; $\mathcal{I}Fml_1(k) := \text{Append}(\mathcal{I}Fml_0(k), A)$ end
4: $\exists xB$: $\mathcal{A}ct_{\text{Ind}} := \mathcal{A}ct_{\text{Ind}} + 1$; extend every non-axiom leaf of S by applying the $(\exists_{\text{Ind}} \rightarrow)$ rule using $y = \text{First}(\mathcal{I}nd_{\mathcal{A}ct_{\text{Ind}}}^{\text{Av}})$ as the new variable; $\mathcal{I}nd_1 := \text{Append}(\mathcal{I}nd_0, [y, []])$ $\mathcal{I}nd_{\mathcal{A}ct_{\text{Ind}}}^{\text{Av}} := \text{Tail}(\mathcal{I}nd_{\mathcal{A}ct_{\text{Ind}}}^{\text{Av}})$
5: $\forall \bar{x}B$: for every tuple $\langle t_1^k, \dots, t_{n_k}^k \rangle \in \text{tup}(\mathcal{T}up_0)$ such that $A \notin \mathcal{S}Fml_0(k)$ do extend every non-axiom leaf of S by applying the $(\forall_{\text{Seq}} \rightarrow)$ rule using the sequence $\ulcorner t_1^k, \dots, t_{n_k}^k \urcorner$ as one of the sequences of the rule; $\mathcal{S}Fml_1(k) := \text{Append}(\mathcal{S}Fml_0(k), A)$ end
6: $\exists \bar{x}B$: $\mathcal{A}ct_{\text{Tup}} := \mathcal{A}ct_{\text{Tup}} + 1$; extend every non-axiom leaf of S by applying the $(\exists_{\text{Seq}} \rightarrow)$ rule using \bar{y} as the new variable where $\langle \bar{y} \rangle = \text{First}(\mathcal{T}up_{\mathcal{A}ct_{\text{Tup}}}^{\text{Av}})$; $\mathcal{T}up_1 := \text{Append}(\mathcal{T}up_0, [\langle \bar{y} \rangle, []])$ $\mathcal{T}up_{\mathcal{A}ct_{\text{Tup}}}^{\text{Av}} := \text{Tail}(\mathcal{T}up_{\mathcal{A}ct_{\text{Tup}}}^{\text{Av}})$
7: end	

Fig. 3. Procedure Grow-Left.

Procedure Grow-Right(A : formula; **var** S : tree)

1: **case** A **of**
2: $B \wedge C, B \vee C$
 $B \Rightarrow C, \neg B$: expand every non-axiom leaf
of S using the right rule
corresponding to the main
propositional connectives;
3: $\exists xB$: **for** every term $t_k \in it(\mathcal{I}nd_0)$
such that $A \notin \mathcal{I}Fml_0(k)$ **do**
extend every non-axiom leaf of S
by applying the $(\rightarrow \exists_{\text{Ind}})$ rule
using the term t_k as one of the
terms of the rule;
 $\mathcal{I}Fml_1(k) := \text{Append}(\mathcal{I}Fml_0(k), A)$
end
4: $\forall xB$: $\mathcal{A}ct_{\text{Ind}} := \mathcal{A}ct_{\text{Ind}} + 1$;
extend every non-axiom leaf of S
by applying the $(\rightarrow \forall_{\text{Ind}})$ rule
using $y = \text{First}(\mathcal{I}nd_{\mathcal{A}ct_{\text{Ind}}}^{\text{Av}})$
as the new variable;
 $\mathcal{I}nd_1 := \text{Append}(\mathcal{I}nd_0, [y, \square])$
 $\mathcal{I}nd_{\mathcal{A}ct_{\text{Ind}}}^{\text{Av}} := \text{Tail}(\mathcal{I}nd_{\mathcal{A}ct_{\text{Ind}}}^{\text{Av}})$
5: $\exists \bar{x}B$: **for** every tuple $\langle t_1^k, \dots, t_{n_k}^k \rangle \in \text{tup}(\mathcal{T}up_0)$
such that $A \notin \mathcal{S}Fml_0(k)$ **do**
extend every non-axiom leaf of S
by applying the $(\rightarrow \exists_{\text{Seq}})$ rule
using the sequence $\lceil t_1^k, \dots, t_{n_k}^k \rceil$ as one
of the sequences of the rule;
 $\mathcal{S}Fml_1(k) := \text{Append}(\mathcal{S}Fml_0(k), A)$
end
6: $\forall \bar{x}B$: $\mathcal{A}ct_{\text{Tup}} := \mathcal{A}ct_{\text{Tup}} + 1$;
extend every non-axiom leaf of S
by applying the $(\rightarrow \forall_{\text{Seq}})$ rule
using \bar{y} as the new variable
where $\langle \bar{y} \rangle = \text{First}(\mathcal{T}up_{\mathcal{A}ct_{\text{Tup}}}^{\text{Av}})$;
 $\mathcal{T}up_1 := \text{Append}(\mathcal{T}up_0, [\langle \bar{y} \rangle, \square])$
 $\mathcal{T}up_{\mathcal{A}ct_{\text{Tup}}}^{\text{Av}} := \text{Tail}(\mathcal{T}up_{\mathcal{A}ct_{\text{Tup}}}^{\text{Av}})$
7: **end**

Fig. 4. Procedure Grow-Right.

$$\begin{aligned}
& \cup \bigcup_{i=0}^{\mathcal{Act}_{\text{Ind}}} \mathcal{I}nd_i^{\text{Av}} \\
& \cup \{s \mid s \in \langle t_1, \dots, t_n \rangle \text{ for some } \langle t_1, \dots, t_n \rangle \in \text{tup}(\mathcal{T}up_0)\} \\
& \cup \bigcup_{i=0}^{\mathcal{Act}_{\text{Tup}}} \{s \mid s \in \langle t_1, \dots, t_n \rangle \text{ for some } \langle t_1, \dots, t_n \rangle \in \mathcal{T}up_i^{\text{Av}}\}
\end{aligned}$$

then $\Gamma_0 \rightarrow \Delta_0$ is falsifiable in a structure with the countable domain H ; or
2. Procedure **Search** generates an infinite tree T and, if we let

$$\begin{aligned}
H = & \text{it}(\mathcal{I}nd_0) \\
& \cup \{s \mid s \in \langle t_1, \dots, t_n \rangle \text{ for some } \langle t_1, \dots, t_n \rangle \in \text{tup}(\mathcal{T}up_0)\},
\end{aligned}$$

then $\Gamma_0 \rightarrow \Delta_0$ is falsifiable in a structure with the countable domain H .

Proof. Assume $\Gamma_0 \rightarrow \Delta_0$ is falsifiable and T be a tree generated by the **Search** procedure. If T was closed, then by Theorem 1 $\Gamma_0 \rightarrow \Delta_0$ would be valid, a contradiction. Thus, the tree T is either infinite, or a finite one containing a finished non-closed leaf.

First we show that H is a term algebra over \mathcal{L}' . Observe that the following three statements hold:

1. $\text{it}(\mathcal{I}nd_0)$, $\mathcal{I}nd_0^{\text{Av}}$, $\text{tup}(\mathcal{T}up_0)$ and $\mathcal{T}up_0^{\text{Av}}$ are initialized in such a way that the variables free in the input sequent belong to the set

$$\begin{aligned}
& \text{it}(\mathcal{I}nd_0) \cup \mathcal{I}nd_0^{\text{Av}} \\
& \cup \{s \mid s \in \langle t_1, \dots, t_n \rangle \text{ for some } \langle t_1, \dots, t_n \rangle \in \text{tup}(\mathcal{T}up_0)\} \\
& \cup \{s \mid s \in \langle t_1, \dots, t_n \rangle \text{ for some } \langle t_1, \dots, t_n \rangle \in \mathcal{T}up_0^{\text{Av}}\}.
\end{aligned}$$

2. Whenever a new individual variable y_i is removed from the head of the list $\mathcal{I}nd_i^{\text{Av}}$, it is added to $\text{it}(\mathcal{I}nd_0)$, and whenever a tuple $\langle \bar{y}_i \rangle$ with a new sequence variable \bar{y}_i is removed from the head of the list $\mathcal{T}up_i^{\text{Av}}$, it is added to $\text{tup}(\mathcal{T}up_0)$.
3. At the end of every round the head of every list $\mathcal{I}nd_i^{\text{Av}}$ for $0 \leq i \leq \mathcal{Act}_{\text{Ind}}$ is removed from $\mathcal{I}nd_i^{\text{Av}}$ and added to $\text{it}(\mathcal{I}nd_0)$. Similarly, at the end of every round, the head of every list $\mathcal{T}up_i^{\text{Av}}$ for $0 \leq i \leq \mathcal{Act}_{\text{Tup}}$ is removed from $\mathcal{T}up_i^{\text{Av}}$ and added to $\text{tup}(\mathcal{T}up_0)$.

If T is finite, from (1), (2) and (3) it follows that the set

$$\begin{aligned}
H = & \text{it}(\mathcal{I}nd_0) \\
& \cup \bigcup_{i=0}^{\mathcal{Act}_{\text{Ind}}} \mathcal{I}nd_i^{\text{Av}} \\
& \cup \{s \mid s \in \langle t_1, \dots, t_n \rangle \text{ for some } \langle t_1, \dots, t_n \rangle \in \text{tup}(\mathcal{T}up_0)\} \\
& \cup \bigcup_{i=0}^{\mathcal{Act}_{\text{Tup}}} \{s \mid s \in \langle t_1, \dots, t_n \rangle \text{ for some } \langle t_1, \dots, t_n \rangle \in \mathcal{T}up_i^{\text{Av}}\}
\end{aligned}$$

contains the set of all terms built up from the variables free in the input sequent, the variables activated during applications of $(\exists_{\text{Ind}} \rightarrow)$, $(\exists_{\text{Seq}} \rightarrow)$, $(\rightarrow \forall_{\text{Ind}})$ or $(\rightarrow \forall_{\text{Seq}})$ rules, and the function symbols occurring in the input sequent. Hence, H is closed under the function symbols in \mathcal{L}' , and it is a term algebra.

If T is infinite, all the terms in all the activated lists $\mathcal{I}nd_i^{\text{Av}}$ are eventually transferred to $it(\mathcal{I}nd_0)$, and all the tuples of terms in all the activated lists $\mathcal{T}up_i^{\text{Av}}$ are eventually transferred to $tup(\mathcal{T}up_0)$. Then the conditions (1) and (3) imply that the set

$$H = it(\mathcal{I}nd_0) \cup \{s \mid s \in \langle t_1, \dots, t_n \rangle \text{ for some } \langle t_1, \dots, t_n \rangle \in tup(\mathcal{T}up_0)\},$$

contains the set of all terms built up from the variables free in the input sequent, the variables activated during applications of $(\exists_{\text{Ind}} \rightarrow)$, $(\exists_{\text{Seq}} \rightarrow)$, $(\rightarrow \forall_{\text{Ind}})$ or $(\rightarrow \forall_{\text{Seq}})$ rules, and the function symbols occurring in the input sequent. Hence, H is a term algebra.

If T is infinite, by König's lemma, it contains an infinite path. In T is finite, it contains a path to a non-axiom finished leaf. In either case we show that a Hintikka set can be found along the path. Let U be the union of all formulae occurring in the left-hand side of each sequent along that path, and V be the union of all formulae occurring in the right-hand side of any such sequent. Let $S = \{\text{TA} \mid A \in U\} \cup \{\text{FA} \mid A \in V\}$. We want to show that S is the Hintikka set with respect to H .

We will show that the conditions *H0-H7* of Definition 25 hold for S .

H0 holds. Since every atomic formula occurring in a sequent occurs in every path having this sequent as source, if S contains both TA and FA for some atom A , then some sequent in the path is an axiom. This contradicts the fact that the path is either infinite, or finite and non-closed.

H1 and *H2* hold. This is true because the definition of α -components and β -components mirrors the inference rules. Since all non-atomic propositions in a sequent $\Gamma \rightarrow \Delta$ on the chosen path are considered during the expansion phase, and since every proposition in the input sequent is eventually considered (as $\text{Head}(L)$ or $\text{Head}(R)$), the following statements hold:

- For every formula $A \in U$, if A belongs to $\Gamma \rightarrow \Delta$ and TA is of type α , then its components α_1 and α_2 are added to the successor of $\Gamma \rightarrow \Delta$ during the expansion step. More precisely, if α_1 (or α_2) is of the form TC_1 (or TC_2), C_1 (C_2) is added to the premise of the successor of $\Gamma \rightarrow \Delta$; if α_1 (or α_2) is of the form FC_1 (or FC_2), C_1 (C_2) is added to the conclusion of the successor of $\Gamma \rightarrow \Delta$; In both cases, α_1 and α_2 belong to S .
- If A belongs to $\Gamma \rightarrow \Delta$ and TA is of type β , its component β_1 is added to the left successor of $\Gamma \rightarrow \Delta$, and β_2 is added to the right successor of $\Gamma \rightarrow \Delta$, during the expansion step. More precisely, if β_1 (or β_2) is of the form TC_1 (or TC_2), C_1 (C_2) is added to the premise of the left (right) successor of $\Gamma \rightarrow \Delta$; if β_1 (or β_2) is of the form FC_1 (or FC_2), C_1 (C_2) is added to the conclusion of the right (left) successor of $\Gamma \rightarrow \Delta$. Hence, either β_1 or β_2 belongs to S .

H3 holds. Every time when a formula C of type γ_i is expanded, all substitution instances $C(t)$ for all t in $it(\mathcal{I}nd_0)$ which have not already been used with C are added to the upper sequent.

H4 holds. Every time when a formula C of type γ_s is expanded, all substitution instances $C(s_1, \dots, s_n)$ for all tuples $\langle s_1, \dots, s_n \rangle$ in $tup(\mathcal{T}up_0)$ which have not already been used with C are added to the upper sequent.

H5 holds. Every time a formula D of type δ_i is expanded, the new individual variable y_i removed from $\mathcal{I}nd_i^{Av}$ is added to $it(\mathcal{I}nd_0)$ and the substitution instance $D(y_i)$ is added to the upper sequent.

H6 holds. Every time a formula D of type δ_s is expanded, the tuple $\langle \bar{y}_i \rangle$ with the new sequence variable \bar{y}_i removed from $\mathcal{T}up_i^{Av}$ is added to $tup(\mathcal{T}up_0)$ and the substitution instance $D(\bar{y}_i)$ is added to the upper sequent.

H7 holds. The propositions (1), (2) and (3) above imply that all variables free in S are in H .

Thus, S is a Hintikka set. By Lemma 8, some assignment $\sigma^{\mathfrak{H}_S}$ satisfies S in a structure \mathfrak{H}_S with the domain H , and, thus, $\Gamma_0 \rightarrow \Delta_0$ is falsified by \mathfrak{H}_S and $\sigma^{\mathfrak{H}_S}$. \square

Lemma 10. *If the input sequent $\Gamma_0 \rightarrow \Delta_0$ is valid, then the procedure **Search** halts with a closed tree T , from which a proof tree for a finite subsequent*

$$C_1, \dots, C_m \rightarrow D_1, \dots, D_n$$

of $\Gamma_0 \rightarrow \Delta_0$ can be constructed.

Proof. Let $\Gamma_0 \rightarrow \Delta_0$ be valid. Assume by contradiction that **Search** does not stop with a closed tree. Then the output of **Search** is either an infinite tree, or a finite tree with a finished non-closed leaf. But in either case, as in the proof of Lemma 9, we can falsify $\Gamma_0 \rightarrow \Delta_0$ by constructing the relevant Hintikka set, which is a contradiction.

A closed tree T that the **Search** procedure returns is not exactly a proof tree. Now we modify it to obtain a proof tree.

Let C_1, \dots, C_m be the initial subsequence of formulae in Γ_0 which were deleted from Γ_0 to obtain L , and D_1, \dots, D_n be the initial subsequent of formulae in Δ_0 which were deleted from Δ_0 to obtain R . A proof tree for a finite sequent $C_1, \dots, C_m \rightarrow D_1, \dots, D_n$ can be obtained from T , using the following technique:

First, starting from the root and proceeding bottom-up, for each node of the tree $\Gamma, \mathbf{Head}(L) \rightarrow \Delta, \mathbf{Head}(R)$ at depth k created at the end of a call to procedure **Expand**, add $\mathbf{Head}(L)$ after the rightmost formula in the premise of every sequent at depth less than k , and add $\mathbf{Head}(R)$ after the rightmost formula in the conclusion of every sequent at depth less than k , obtaining the tree T' . Then, a proof tree T'' for $C_1, \dots, C_m \rightarrow D_1, \dots, D_n$ is constructed from T' by deleting all duplicate nodes. The tree T'' is a proof tree because the same inference rules that have been used in T' are used in T'' . \square

From Lemma 10 and Theorem 1 we get the following version of Gödel's extended completeness theorem for the calculus \mathcal{G} :

Theorem 2 (Completeness of \mathbf{G}). *A sequent (even infinite) is valid iff it is provable in \mathbf{G} .*

For quantifier-free formulae we have the following useful result:

Theorem 3. *There is an algorithm for deciding whether a finite sequent consisting of quantifier-free formulae is valid.*

The following result shows that the Löwenheim-Skolem theorem is valid for the language \mathcal{L} :

Theorem 4 (Löwenheim-Skolem, for \mathcal{L}). *If a set of formulae Γ of the language \mathcal{L} is satisfiable in some structure \mathfrak{S} , then it is satisfiable in a structure whose domain is at most countably infinite.*

Proof. It is clear that $\Gamma \rightarrow$ is falsifiable by \mathfrak{S} . By Lemma 9, the **Search** procedure yields a tree from which a Hintikka set S can be obtained. But the domain \mathcal{H} of \mathfrak{H}_S is at most countably infinite since it consists of terms built from countable sets. Hence, \mathfrak{H}_S is a countable structure in which Γ is satisfiable. \square

Theorem 5 (Compactness, for \mathcal{L}). *For any (possibly countably infinite) set Γ of formulae of the language \mathcal{L} , if every non-empty finite subset of Γ is satisfiable then Γ is satisfiable.*

Proof. Let every non-empty finite subset of Γ be satisfiable. Assume by contradiction that Γ is unsatisfiable. Viewing Γ as a sequence of formulae, the sequent $\Gamma \rightarrow$ is valid, and by Lemma 10 there exists a finite subsequence F_1, \dots, F_m of Γ such that $F_1, \dots, F_m \rightarrow$ is provable in \mathbf{G} . But then, by Soundness Theorem, $F_1, \dots, F_m \rightarrow$ is valid, which means that F_1, \dots, F_m is not satisfiable. It contradicts the assumption that every non-empty finite subset of Γ is satisfiable. The obtained contradiction proves that Γ is satisfiable. \square

Definition 27. *A set Γ of formulae is consistent if there exists some formula A such that $F_1, \dots, F_m \rightarrow A$ is not provable for any $F_1, \dots, F_m \rightarrow A$ in Γ .*

Theorem 6 (Model Existence Theorem, for \mathcal{L}). *If a set of formulae Γ of the language \mathcal{L} is consistent then it is satisfiable.*

Proof. Let Γ be consistent. Assume by contradiction that Γ is unsatisfiable. Hence, for every formula B , the sequent $\Gamma \rightarrow B$ is valid. By Lemma 10 there exists a finite subsequence F_1, \dots, F_m of Γ such that the sequent $F_1, \dots, F_m \rightarrow B$ is provable. But the Γ is not consistent, a contradiction. \square

The converse of Model Existence Theorem is also true.

Lemma 11 (Consistency Lemma, for \mathcal{L}). *If a set of formulae Γ of the language \mathcal{L} is satisfiable then it is consistent.*

Proof. Let \mathfrak{S} be a structure such that $\mathfrak{S} \models \Gamma$. Assume by contradiction that Γ is inconsistent. Then $\Gamma \rightarrow A$ is provable for every formula A , and, in particular, there is a finite subsequence F_1, \dots, F_m of Γ such that $F_1, \dots, F_m \rightarrow B \wedge \neg B$ is provable for some sentence B . By Soundness Theorem, $F_1, \dots, F_m \rightarrow B \wedge \neg B$ is valid and, since we have $\mathfrak{S} \models \Gamma$, we obtain that $\mathfrak{S} \models B \wedge \neg B$, which is impossible. Hence, Γ is consistent. \square

8 The Gentzen-Style System \mathbf{G}^{\approx}

We define a Gentzen-Style System \mathbf{G}^{\approx} for the language with equality \mathcal{L}_{\approx} .

Below the symbols Γ, Δ, A will be used to denote arbitrary sequences of formulae. Position is defined as follows:

Definition 28. A position within a term or atom E is a sequence of positive integers, describing the path from $\text{Head}(E)$ to the head of the subterm at that position. By $E[s]_p$ we denote the term or atom obtained from E by replacing the term at position p with the term s .

Definition 29. The sequent calculus \mathbf{G}^{\approx} consists of the inference rules of the calculus \mathbf{G} and, in addition, the following equality rule:

$$\frac{\Gamma, (s \approx t \wedge A[s]_p) \Rightarrow A[t]_p \rightarrow \Delta}{\Gamma \rightarrow \Delta} (\approx)$$

where s, t are terms, A is an atom, and p is a position in A .

The axioms of \mathbf{G}^{\approx} are axioms of \mathbf{G} and, in addition, sequents of the form $\Gamma \rightarrow \Delta, s \approx s, A$, called the *reflexivity axiom* for \approx .

9 Soundness of \mathbf{G}^{\approx}

Soundness of \mathbf{G}^{\approx} is easy to show.

Lemma 12. *The reflexivity axiom is valid.*

Proof. The lemma follows from the fact that for any individual term s , the formula $s \approx s$ is valid. \square

Lemma 13. *The conclusion of the \approx rule in Definition 29 is valid iff its premise is valid.*

Proof. Since the formula $(s \approx t \wedge A[s]_p) \Rightarrow A[t]_p$ is valid, proof of the lemma is straightforward. \square

These lemmata and soundness of \mathbf{G} imply soundness of \mathbf{G}^{\approx} :

Theorem 7 (Soundness of \mathbf{G}^{\approx}). *Every sequent provable in \mathbf{G}^{\approx} is valid.*

10 Completeness of G^{\approx}

10.1 Hintikka Sets for Languages with Equality

In this subsection we generalize the concept of a Hintikka set for languages with equality.

Definition 30. A Hintikka set E over a language \mathcal{L}_{\approx} with equality with respect to a term algebra \mathcal{H} (over the reduct \mathcal{L}_{\approx_E}) is a set of signed \mathcal{L}_{\approx} -formulae such that the following conditions hold for all signed formulae A, B, C, D of type α, β, γ and δ :

- H0 No atomic formula and its conjugate are both in E .
- H1 If a type- α formula A is in E , then both A_1 and A_2 are in E .
- H2 If a type- β formula B is in E , then either B_1 is in E or B_2 is in E .
- H3 If a type- γ_i formula C is in E , then for every $t \in \mathcal{H}_{\text{Ind}}$, $C(t)$ is in E (we require that t is free for x in C for every $t \in \mathcal{H}_{\text{Ind}}$).
- H4 If a type- γ_s formula C is in E , then for every $n \geq 0$ and terms $s_1, \dots, s_n \in \mathcal{H}$, $C(s_1, \dots, s_n)$ is in E (we require that s_1, \dots, s_n are free for \bar{x} in C for every $s_1, \dots, s_n \in \mathcal{H}$).
- H5 If a type- δ_i formula D is in E , then there exists a $t \in \mathcal{H}_{\text{Ind}}$ such that $D(t)$ is in E (we require that t is free for x in D for every $t \in \mathcal{H}_{\text{Ind}}$).
- H6 If a type- δ_s formula D is in E , then there exist terms $s_1, \dots, s_n \in \mathcal{H}$ such that $D(s_1, \dots, s_n)$ is in E (we require that s_1, \dots, s_n are free for \bar{x} in D for every $s_1, \dots, s_n \in \mathcal{H}$).
- H7 Every (individual or sequence) variable v occurring free in some formula of E is in \mathcal{H} .
- H8 (a) For every $t \in \mathcal{H}_{\text{Ind}}$, $\top(t \approx t) \in E$.
 (b) For all $s, t \in \mathcal{H}_{\text{Ind}}$, an atom $A \in \mathcal{L}_{\approx}$, and a position p in A , $\top((s \approx t \wedge A[s]_p) \Rightarrow A[t]_p) \in E$.

We assume without loss of generality that the set of variables occurring free in formulae in E is disjoint from the set of variables occurring bound in formulae in E .

Our goal is to prove that any Hintikka set over a language with equality is satisfiable. For this we need additional definitions and propositions:

Definition 31. A binary relation R on the set of terms $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is called congruence if R satisfies the following properties:

1. Reflexivity: for all $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, tRt .
2. Symmetry: for all $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, if sRt , then tRs .
3. Transitivity: for all $s, t, r \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, if sRr and rRt , then sRt .
4. Monotonicity: for all $s, t, r \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ and positions p , if sRt , then $r[s]_pRr[t]_p$.

Definition 32. Let E be a Hintikka set (over a language with equality) with respect to a term algebra $\mathcal{H} = \mathcal{H}_{\text{Ind}} \uplus \mathcal{H}_{\text{Seq}}$. Then by $\cong_{\mathcal{H}_{\text{Ind}}}^E$ we denote a binary relation defined on \mathcal{H}_{Ind} as follows: For all $s, t \in \mathcal{H}_{\text{Ind}}$, $s \cong_{\mathcal{H}_{\text{Ind}}}^E t$ iff $\top(s \approx t) \in E$.

Lemma 14. *Let E be a Hintikka set (over a language with equality) with respect to a term algebra $\mathcal{H} = \mathcal{H}_{\text{Ind}} \uplus \mathcal{H}_{\text{Seq}}$. Then $\cong_{\mathcal{H}_{\text{Ind}}}^E$ is a congruence on \mathcal{H}_{Ind} .*

Proof. 1. Reflexivity follows from $H8(a)$.

2. Symmetry. Assume $s \cong_{\mathcal{H}_{\text{Ind}}}^E t$. Then by the definition of $\cong_{\mathcal{H}_{\text{Ind}}}^E$, $\top(s \approx t) \in E$. By $H8(b)$, $\top((s \approx t \wedge s \approx s) \Rightarrow t \approx s) \in E$. Then, by $H2$, either $\text{F}(s \approx t \wedge s \approx s) \in E$ or $\top(t \approx s) \in E$. Since $\text{F}(s \approx t \wedge s \approx s) \in E$ leads to a contradiction, we must have $\top(t \approx s) \in E$ and, therefore, $t \cong_{\mathcal{H}_{\text{Ind}}}^E s$.
3. Transitivity. Assume $s \cong_{\mathcal{H}_{\text{Ind}}}^E r$ and $r \cong_{\mathcal{H}_{\text{Ind}}}^E t$. Then, by the definition of $\cong_{\mathcal{H}_{\text{Ind}}}^E$, $\top(s \approx r) \in E$ and $\top(r \approx t) \in E$. By $H8(b)$, $\top((r \approx t \wedge s \approx r) \Rightarrow s \approx t) \in E$. By $H2$, $\text{F}(r \approx t \wedge s \approx r) \in E$ or $\top(s \approx t) \in E$. The first alternative leads to a contradiction, therefore we must have $\top(s \approx t) \in E$. Hence, $s \cong_{\mathcal{H}_{\text{Ind}}}^E t$.
4. Monotonicity. Assume $s \cong_{\mathcal{H}_{\text{Ind}}}^E t$. By reflexivity, $r[s]_p \cong_{\mathcal{H}_{\text{Ind}}}^E r[s]_p$. By the definition of $\cong_{\mathcal{H}_{\text{Ind}}}^E$, $\top(s \approx t) \in E$ and $\top(r[s]_p \approx r[s]_p) \in E$. By $H8(b)$, $\top((s \approx t \wedge r[s]_p \approx r[s]_p) \Rightarrow r[s]_p \approx r[t]_p) \in E$. By $H2$, $\text{F}(s \approx t \wedge r[s]_p \approx r[s]_p) \in E$ or $\top(r[s]_p \approx r[t]_p) \in E$. The first alternative leads to a contradiction, therefore we must have $\top(r[s]_p \approx r[t]_p) \in E$. Hence, $r[s]_p \cong_{\mathcal{H}_{\text{Ind}}}^E r[t]_p$. \square

Lemma 15. *Let E be a Hintikka set (over a language with equality) with respect to a term algebra $\mathcal{H} = \mathcal{H}_{\text{Ind}} \uplus \mathcal{H}_{\text{Seq}}$. Then for any terms $s, t \in \mathcal{H}_{\text{Ind}}$ such that $s \cong_{\mathcal{H}_{\text{Ind}}}^E t$, an atom $A \in \mathcal{L}_{\approx E}$, and a position p in A ,*

1. $\top(A[s]_p) \in E$ iff $\top(A[t]_p) \in E$.
2. $\text{F}(A[s]_p) \in E$ iff $\text{F}(A[t]_p) \in E$.

Proof. 1. (\Rightarrow) First assume $\top(A[s]_p) \in E$. By $H8(b)$, $\top((s \approx t \wedge A[s]_p) \Rightarrow A[t]_p) \in E$. Then by $H2$ we get either $\text{F}(s \approx t \wedge A[s]_p) \in S$ or $\top(A[t]_p) \in E$. The first alternative leads to a contradiction. Therefore, we must have $\top(A[t]_p) \in E$.

(\Leftarrow) Now assume $\top(A[t]_p) \in S$. Then by symmetry of \cong , and the same reasoning as in the direction (\Rightarrow) above we obtain that $\top(A[s]_p) \in S$.

2. (\Rightarrow) First assume $\text{F}(A[s]_p) \in E$. By $H8(b)$, $\top((t \approx s \wedge A[t]_p) \Rightarrow A[s]_p) \in E$. Then by $H2$ we get either $\text{F}(t \approx s \wedge A[t]_p) \in S$ or $\top(A[s]_p) \in E$. The second alternative immediately leads to a contradiction. From the first alternative we get either $\text{F}(t \approx s) \in E$ or $\text{F}(A[t]_p) \in E$. By the symmetry and the definition of \cong , $\text{F}(t \approx s) \in E$ gives a contradiction as well. Hence, we must have $\text{F}(A[t]_p) \in E$.

(\Leftarrow) Now assume $\text{F}(A[t]_p) \in S$. By $H8(b)$, $\top((s \approx t \wedge A[s]_p) \Rightarrow A[t]_p) \in E$. By $H2$, either $\text{F}(s \approx t \wedge A[s]_p) \in S$ or $\top(A[t]_p) \in E$. The second alternative immediately leads to a contradiction. From the first alternative we get either $\text{F}(s \approx t) \in E$ or $\text{F}(A[s]_p) \in E$. But $\text{F}(s \approx t) \in E$ gives a contradiction as well. Hence, we must have $\text{F}(A[s]_p) \in E$. \square

Now we extend $\cong_{\mathcal{H}_{\text{Ind}}}^E$ to \mathcal{H} :

Definition 33. *Let E be a Hintikka set (over a language with equality) with respect to a term algebra $\mathcal{H} = \mathcal{H}_{\text{Ind}} \uplus \mathcal{H}_{\text{Seq}}$. Then by $\cong_{\mathcal{H}}^E$ we denote a binary relation defined on \mathcal{H} as follows:*

1. For all $s, t \in \mathcal{H}$, if $s \cong_{\mathcal{H}_{\text{Ind}}}^E t$, then $s \cong_{\mathcal{H}}^E t$.
2. For every $s \in \mathcal{H}$, $s \cong_{\mathcal{H}}^E s$.
3. For all $s, t, r, q \in \mathcal{H}$ and position p , if $s \cong_{\mathcal{H}}^E t$ and $r[s]_p \cong_{\mathcal{H}}^E q$, then $r[t]_p \cong_{\mathcal{H}}^E q$.

Lemma 16. *Let E be a Hintikka set (over a language with equality) with respect to a term algebra $\mathcal{H} = \mathcal{H}_{\text{Ind}} \cup \mathcal{H}_{\text{Seq}}$. Then $\cong_{\mathcal{H}}^E$ is a congruence on \mathcal{H} which coincides with $\cong_{\mathcal{H}_{\text{Ind}}}^E$ on \mathcal{H}_{Ind} .*

Proof. 1. Reflexivity follows from the definition of $\cong_{\mathcal{H}}^E$.
2. Symmetry. Assume $s \cong_{\mathcal{H}}^E t$. By reflexivity, $s \cong_{\mathcal{H}}^E s$. Then by the condition 3 of the definition of $\cong_{\mathcal{H}}^E$ we get $t \cong_{\mathcal{H}}^E s$.
3. Transitivity. Assume $s \cong_{\mathcal{H}}^E r$ and $r \cong_{\mathcal{H}}^E t$. By symmetry, $r \cong_{\mathcal{H}}^E s$. By the condition 3 of the definition of $\cong_{\mathcal{H}}^E$ we get $s \cong_{\mathcal{H}}^E t$.
4. Monotonicity. Assume $s \cong_{\mathcal{H}}^E t$. By reflexivity, $r[s]_p \cong_{\mathcal{H}}^E r[s]_p$. By the condition 3 of the definition of $\cong_{\mathcal{H}}^E$, $r[t]_p \cong_{\mathcal{H}}^E r[s]_p$. By symmetry, $r[s]_p \cong_{\mathcal{H}}^E r[t]_p \in E$.

The fact that $\cong_{\mathcal{H}}^E$ coincides with $\cong_{\mathcal{H}_{\text{Ind}}}^E$ on \mathcal{H}_{Ind} follows immediately from the condition 1 of the definition of $\cong_{\mathcal{H}}^E$. \square

The following counterpart of Lemma 15 holds:

Lemma 17. *Let E be a Hintikka set (over a language with equality) with respect to a term algebra $\mathcal{H} = \mathcal{H}_{\text{Ind}} \cup \mathcal{H}_{\text{Seq}}$. Then for any terms $s, t \in \mathcal{H}$ such that $s \cong_{\mathcal{H}}^E t$, an atom $A \in \mathcal{L}_{\approx E}$, and a position p in A*

1. $\text{T}(A[s]_p) \in E$ iff $\text{T}(A[t]_p) \in E$.
2. $\text{F}(A[s]_p) \in E$ iff $\text{F}(A[t]_p) \in E$.

Proof. If s and t both are individual terms, then Lemma 15 implies the result. Assume s and t both are sequence terms. Then $s \cong_{\mathcal{H}}^E t$ implies either $s = t$, or there exist positions p_1, \dots, p_n in s and t such that $s_i \cong_{\mathcal{H}_{\text{Ind}}}^E t_i$ for all $1 \leq i \leq n$, where $s_i = s|_{p_i}$ and $t_i = t|_{p_i}$ are individual terms. Moreover, for all $1 \leq i, j \leq n$, $i \neq j$ implies that p_i is not a prefix of p_j .

For the first case, when $s = t$, the lemma is trivial. We prove it for the second case. We denote by A_0 the atom $A[s]_p$, and for all $1 \leq i \leq n$, by A_i the atom $A_{i-1}[t_i]_{p.p_i}$. By Lemma 15 we have $\text{T}(A_{i-1}[s_i]_{p.p_i}) \in E$ iff $\text{T}(A_{i-1}[t_i]_{p.p_i}) \in E$ and $\text{F}(A_{i-1}[s_i]_{p.p_i}) \in E$ iff $\text{F}(A_{i-1}[t_i]_{p.p_i}) \in E$ for all $1 \leq i \leq n$. It implies that $\text{T}(A_0[s_1]_{p.p_1}) \in E$ iff $\text{T}(A_{n-1}[t_n]_{p.p_n}) \in E$ and $\text{F}(A_0[s_1]_{p.p_1}) \in E$ iff $\text{F}(A_{n-1}[t_n]_{p.p_n}) \in E$. By construction, $A_0[s_1]_{p.p_1} = A[s]_p$ and $A_{n-1}[t_n]_{p.p_n} = A[t]_p$, which finishes the proof. \square

Now we prove a generalization of Lemma 8 for languages with equality.

Lemma 18. *Every Hintikka set E (over a language with equality) with respect to a term algebra \mathcal{H} is satisfiable in a structure \mathfrak{S}_E with the domain $\mathcal{H}_E = \mathcal{H} / \cong_{\mathcal{H}}^E$, where $\mathcal{H} / \cong_{\mathcal{H}}^E$ is a quotient of \mathcal{H} modulo the relation $\cong_{\mathcal{H}}^E$.*

Proof. We define the structure $\mathfrak{H}_E = \langle \mathcal{H}_E, \mathcal{I} \rangle$ as follows. The domain \mathcal{H}_E of this structure is the quotient of the set \mathcal{H} modulo the equivalence relation $\cong_{\mathcal{H}}^E$. Then $\mathcal{H}_{E_{\text{Ind}}} = \mathcal{H}_{\text{Ind}} / \cong_{\mathcal{H}}^E$ and $\mathcal{H}_{E_{\text{Seq}}} = \mathcal{H}_{\text{Seq}} / \cong_{\mathcal{H}}^E$. Given a term $t \in \mathcal{H}$, we denote by $[t]$ its equivalence class. The interpretation \mathcal{I} associates:

- To every individual constant c in $\mathcal{L}_{\approx E}$ its equivalence class $c_{\mathcal{I}} = [c] \in \mathcal{H}_{E_{\text{Ind}}}$.
- To every sequence constant \bar{c} in $\mathcal{L}_{\approx E}$ some element $c_{\mathcal{I}} = [\bar{c}] \in \mathcal{H}_{E_{\text{Seq}}}$.
- To every n -ary individual function symbol f in $\mathcal{L}_{\approx E}$, with $n > 0$, some n -ary function $f_{\mathcal{I}} : \mathcal{H}_{E_{\text{Ind}}}^n \rightarrow \mathcal{H}_{E_{\text{Ind}}}$ such that $f_{\mathcal{I}}([t_1], \dots, [t_n]) = [f_{\mathcal{I}}(t_1, \dots, t_n)]$ for any equivalence classes $[t_1], \dots, [t_n] \in \mathcal{H}_{E_{\text{Ind}}}$.
- To every n -ary sequence function symbol \bar{f} in $\mathcal{L}_{\approx E}$, with $n > 0$, some n -ary function $\bar{f}_{\mathcal{I}} : \mathcal{H}_{E_{\text{Seq}}}^n \rightarrow \mathcal{H}_{E_{\text{Seq}}}$ such that $\bar{f}_{\mathcal{I}}([t_1], \dots, [t_n]) = [\bar{f}_{\mathcal{I}}(t_1, \dots, t_n)]$ for any equivalence classes $[t_1], \dots, [t_n] \in \mathcal{H}_{E_{\text{Seq}}}$.
- To every flexible arity individual function symbol f in $\mathcal{L}_{\approx E}$, some flexible arity function $f_{\mathcal{I}} : \mathcal{H}_E^{\infty} \rightarrow \mathcal{H}_{E_{\text{Ind}}}$ such that $f_{\mathcal{I}}([t_1], \dots, [t_n]) = [f_{\mathcal{I}}(t_1, \dots, t_n)]$ for any equivalence classes $[t_1], \dots, [t_n] \in \mathcal{H}_E$.
- To every flexible arity sequence function symbol \bar{f} in $\mathcal{L}_{\approx E}$, some flexible arity function $\bar{f}_{\mathcal{I}} : \mathcal{H}_E^{\infty} \rightarrow \mathcal{H}_{E_{\text{Seq}}}$, such that $\bar{f}_{\mathcal{I}}([t_1], \dots, [t_n]) = [\bar{f}_{\mathcal{I}}(t_1, \dots, t_n)]$ for any equivalence classes $[t_1], \dots, [t_n] \in \mathcal{H}_E$.
- To every n -ary predicate symbol p in $\mathcal{L}_{\approx E}$, with $n \geq 0$, some n -ary predicate $p_{\mathcal{I}} \subseteq \mathcal{H}_{E_{\text{Ind}}}^n$ such that for any equivalence classes $[t_1], \dots, [t_n] \in \mathcal{H}_{E_{\text{Ind}}}$,

$$p_{\mathcal{I}}([t_1], \dots, [t_n]) = \begin{cases} \mathbf{T}, & \text{if } \top p(t_1, \dots, t_n) \in E \\ \mathbf{F}, & \text{if } \text{F}p(t_1, \dots, t_n) \in E \\ & \text{or neither } \top p(t_1, \dots, t_n) \text{ nor } \text{F}p(t_1, \dots, t_n) \text{ is in } E. \end{cases}$$

- To every flexible arity predicate symbol p in $\mathcal{L}_{\approx E}$ some flexible arity predicate $p_{\mathcal{I}} \subseteq \mathcal{H}_E^{\infty}$ such that for any equivalence classes $[t_1], \dots, [t_n] \in \mathcal{H}_E$,

$$p_{\mathcal{I}}([t_1], \dots, [t_n]) = \begin{cases} \mathbf{T}, & \text{if } \top p(t_1, \dots, t_n) \in E \\ \mathbf{F}, & \text{if } \text{F}p(t_1, \dots, t_n) \in E \\ & \text{or neither } \top p(t_1, \dots, t_n) \text{ nor } \text{F}p(t_1, \dots, t_n) \text{ is in } E. \end{cases}$$

Lemma 16 and Lemma 17 guarantee that interpretation for function and predicate symbols is well defined, i.e., their definition is independent of the particular choice of representatives in the equivalence classes.

Let $\sigma^{\mathfrak{H}_E}$ be a state such that $\sigma^{\mathfrak{H}_E}(v) = [v]$ for every variable $v \in \mathcal{H}$. It remains to show that \mathfrak{H}_E and $\sigma^{\mathfrak{H}_E}$ satisfy E .

For this, first we prove by induction on terms that for every term $t \in \mathcal{H}$, $\mathcal{V}al_{\sigma}^{\mathfrak{H}_E}(t) = [t]$: If t is a variable, $\mathcal{V}al_{\sigma}^{\mathfrak{H}_E}(t) = \sigma^{\mathfrak{H}_E}(t) = [t]$. If t is a constant, then $\mathcal{V}al_{\sigma}^{\mathfrak{H}_E}(t) = t_{\mathcal{I}} = [t]$. If t is a term $f(s_1, \dots, s_n)$, then by the induction hypothesis, $\mathcal{V}al_{\sigma}^{\mathfrak{H}_E}(s_i) = [s_i]$ for all $1 \leq i \leq n$ and, therefore, $\mathcal{V}al_{\sigma}^{\mathfrak{H}_E}(f(s_1, \dots, s_n)) = f_{\mathcal{I}}(\mathcal{V}al_{\sigma}^{\mathfrak{H}_E}(s_1), \dots, \mathcal{V}al_{\sigma}^{\mathfrak{H}_E}(s_n)) = f_{\mathcal{I}}([s_1], \dots, [s_n]) = [f(s_1, \dots, s_n)] = [t]$.

Now, we show that for any atomic formula $\top(p(t_1, \dots, t_n)) \in E$ the truth value $\mathcal{V}al_{\sigma}^{\mathfrak{H}_E}(\top(p(t_1, \dots, t_n))) = \mathbf{T}$. Indeed, if $\top(p(t_1, \dots, t_n)) \in E$, by H7 and since \mathcal{H} is a term algebra, we have $t_1, \dots, t_n \in \mathcal{H}$. Then $\mathcal{V}al_{\sigma}^{\mathfrak{H}_E}(\top(p(t_1, \dots, t_n))) = p_{\mathcal{I}}(\mathcal{V}al_{\sigma}^{\mathfrak{H}_E}(t_1), \dots, \mathcal{V}al_{\sigma}^{\mathfrak{H}_E}(t_n)) = p_{\mathcal{I}}([t_1], \dots, [t_n]) = \mathbf{T}$.

A similar proof argument applies when $F(p(t_1, \dots, t_n)) \in E$. Also, if $\mathbf{T}(t_1 \approx t_2) \in E$, by definition of $\cong_{\mathcal{H}}^E$ we have $t_1 \cong_{\mathcal{H}}^E t_2$, which is equivalent to $[t_1] = [t_2]$, that is, $\mathcal{V}al_{\sigma}^{\mathfrak{H}E}(t_1) = \mathcal{V}al_{\sigma}^{\mathfrak{H}E}(t_2)$. Therefore, $\mathcal{V}al_{\sigma}^{\mathfrak{H}E}(t_1 \approx t_2) = \mathbf{T}$.

The rest of the proof proceeds using the induction principle for formulae. The propositional connectives can be handled as before.

We give now a proof for a signed formula of type γ . If a formula C of type γ_i is in E , then by *H3*, $C(t) \in E$ for every term $t \in \mathcal{H}$. By the induction hypothesis $\mathcal{V}al_{\sigma}^{\mathfrak{H}E}(C(t)) = \mathbf{T}$. By Lemma 6, for any formula F , any individual term t free for x in a F , any structure \mathfrak{S} and any state θ , $\mathcal{V}al_{\theta}^{\mathfrak{S}}(F\{x \mapsto t\}) = \mathcal{V}al_{\theta}^{\mathfrak{S}}(F\{x \mapsto d_{\mathfrak{S}}\})$, where $\mathcal{V}al_{\theta}^{\mathfrak{S}}(t) = d$. Since $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(t) = [t]$, we have $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(C\{x \mapsto t\}) = \mathcal{V}al_{\sigma}^{\mathfrak{S}}(C\{x \mapsto [t]_{\mathfrak{S}}\})$. Hence, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(C(t)) = \mathbf{T}$ iff $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(C([t]_{\mathfrak{S}})) = \mathbf{T}$, and since $[t] \in \mathcal{H}_E$, by Lemma 4, $\mathcal{V}al_{\sigma}^{\mathfrak{S}}(C) = \mathbf{T}$. For a formula of type γ_s the proof is similar.

For a signed formula of type δ we can apply similar arguments.

Finally, using Remark 2, \mathfrak{H}_E can be expanded to an \mathcal{L}_{\approx} -structure satisfying E . It finishes the proof. \square

10.2 Organizing the Terms

To prove the completeness of \mathbf{G}^{\approx} we modify the procedure **Expand** in the following way. Given a sequent $\Gamma_0 \rightarrow \Delta_0$, let \mathcal{L}' be the reduct of \mathcal{L} consisting of all function and predicate symbols occurring in formulae in $\Gamma_0 \rightarrow \Delta_0$.

Let \mathcal{EQ}_0 be an enumeration of all \mathcal{L}' -formulae of the form $s \approx t \wedge A[s]_p \Rightarrow A[t]_p$, where the terms s and t are built from the function symbols in \mathcal{L}' , and the set of variables free in $\Gamma_0 \rightarrow \Delta_0$. The atom A is built from the predicate and function symbols in \mathcal{L}' , and the set of variables free in $\Gamma_0 \rightarrow \Delta_0$. For $i \geq 1$ we define the sets \mathcal{EQ}_i as above, except that the terms s and t belong to the lists $\mathcal{I}nd_i^{\text{Av}}$. During a round, at the end of every expansion step, we shall add each formula that is the head of the list \mathcal{EQ}_i for all $0 \leq i \leq \mathcal{A}ct_{\text{Ind}}$, to the antecedent of every leaf sequent. At the end of each round, each such formula is deleted from the head of the list \mathcal{EQ}_i .

10.3 The Search $_{\approx}$ Procedure

The **Search $_{\approx}$** procedure for sequents containing the equality symbol is shown on Fig. 5.

The notion of *finished leaf*, which occurs on lines 8 and 13 of the procedure **Search $_{\approx}$** differs from the notion of finished leaf in **Search**. Here, a leaf of the tree is finished iff it is an axiom. Hence, the **Search $_{\approx}$** procedure builds an infinite tree if the sequent is not valid.

The procedure **Expand $_{\approx}$** called on line 14 of the procedure **Search $_{\approx}$** is given on Fig. 6.

We now establish counterparts of Lemma 9 and Lemma 10 for sequents containing the equality symbol.

```

Procedure Search≈( $\Gamma_0 \rightarrow \Delta_0$  : sequent; var  $T$  : tree)
1:  $L := \text{Tail}(\Gamma_0)$ ;  $\Gamma := \text{Head}(\Gamma_0)$ 
2:  $R := \text{Tail}(\Delta_0)$ ;  $\Delta := \text{Head}(\Delta_0)$ 
3:  $T :=$  one-node tree labeled with  $\Gamma \rightarrow \Delta$ 
4: initialize  $\text{Up}_0, \text{Up}_i^{\text{Av}}, i \geq 0$ , as above
5: initialize  $\text{Ind}_0, \text{Ind}_i^{\text{Av}}, i \geq 0$ , as above
6: initialize  $\mathcal{EQ}_i, i \geq 0$ , as above
7:  $\text{Act}_{\text{Up}} := 0$ ;  $\text{Act}_{\text{Ind}} := 0$ 
8: while not all leaves of  $T$  are finished do
9:    $\text{Up}_1 := \text{Up}_0$ ;  $\text{Ind}_1 := \text{Ind}_0$ 
10:   $T_0 := T$ ;  $\text{Act}_{\text{Ind}_0} = \text{Act}_{\text{Ind}}$ 
11:  for each leaf node of  $T_0$ 
12:    (in lexicographic order) do
13:      if not finished(node) then
14:        Expand≈(node,  $T$ )
15:      end
16:    end
17:  for  $i := 0$  to  $\text{Act}_{\text{Ind}_0}$  do
18:     $\mathcal{EQ}_i := \text{Tail}(\mathcal{EQ}_i)$ 
19:  end
20:   $\text{Up}_0 := \text{Up}_1$ ;  $\text{Ind}_0 := \text{Ind}_1$ 
21:   $L := \text{Tail}(L)$ ;  $R := \text{Tail}(R)$ 
22:  for  $i := 0$  to  $\text{Act}_{\text{Up}}$  do
23:     $F := [\text{First}(\text{Up}_i^{\text{Av}}), []]$ 
24:     $\text{Up}_0 := \text{Append}(\text{Up}_0, F)$ 
25:     $\text{Up}_i^{\text{Av}} := \text{Tail}(\text{Up}_i^{\text{Av}})$ 
26:  end
27:  for  $i := 0$  to  $\text{Act}_{\text{Ind}}$  do
28:     $F := [\text{First}(\text{Ind}_i^{\text{Av}}), []]$ 
29:     $\text{Ind}_0 := \text{Append}(\text{Ind}_0, F)$ 
30:     $\text{Ind}_i^{\text{Av}} := \text{Tail}(\text{Ind}_i^{\text{Av}})$ 
31:  end
32: end
33: if all leaves are closed then
34:   write(" $T$  is a proof of  $\Gamma_0 \rightarrow \Delta_0$ ")
35: else write(" $\Gamma_0 \rightarrow \Delta_0$  is falsifiable")
36: end

```

Fig. 5. Procedure Search_≈.

```

Procedure  $\text{Expand}_{\approx}(node : \text{tree-address}; \text{var } T : \text{tree})$ 
1: let  $A_1, \dots, A_m \rightarrow B_1, \dots, B_n$  be the label of  $node$ 
2:  $S :=$  one-node tree
   labeled with  $A_1, \dots, A_m \rightarrow B_1, \dots, B_n$ 
3: for  $i := 1$  to  $m$  do
4:   if  $\text{non-atomic}(A_i)$  then
5:      $\text{Grow-Left}(A_i, S)$ 
6:   end
7: end
8: for  $i := 1$  to  $n$  do
9:   if  $\text{non-atomic}(B_i)$  then
10:     $\text{Grow-Right}(B_i, S)$ 
11:   end
12: end
13: for each leaf  $l$  in  $S$  do
14:   let  $\Gamma \rightarrow \Delta$  be a label of  $l$ 
15:    $\Gamma' := \Gamma, \text{First}(L)$ 
16:    $\Delta' := \Delta, \text{First}(R)$ 
17:   for  $i := 0$  to  $\text{Act}_{\text{Ind}_0}$  do
18:      $\Gamma' := \Gamma', \text{First}(\mathcal{E}Q_i)$ 
19:   end
20:   create a new node  $l_1$  labeled with  $\Gamma' \rightarrow \Delta'$ 
21:  $T := \text{substitute}(T, node, S)$ 

```

Fig. 6. Procedure Expand_{\approx} .

Lemma 19. *If the input sequent $\Gamma_0 \rightarrow \Delta_0$ containing the equality symbol \approx is falsifiable, then Procedure Search_{\approx} generates an infinite tree T and, if we let*

$$\begin{aligned}
H = & \text{it}(\mathcal{I}\text{nd}_0) \\
& \cup \{s \mid s \in \langle t_1, \dots, t_n \rangle \text{ for some } \langle t_1, \dots, t_n \rangle \in \text{tup}(\mathcal{T}\text{up}_0)\},
\end{aligned}$$

then $\Gamma_0 \rightarrow \Delta_0$ is falsifiable in a structure with the countable domain which is a quotient of H .

Proof. Assume $\Gamma_0 \rightarrow \Delta_0$ is falsifiable and T be a tree generated by the Search_{\approx} procedure. If T was closed, then by Theorem 7 $\Gamma_0 \rightarrow \Delta_0$ would be valid, a contradiction. Thus, T is not closed, which, by construction of T , means that T is infinite.

We can show that H is a term algebra over \mathcal{L}' as it was done in the proof of Lemma 9.

If T is infinite, by König's lemma, it contains an infinite path. We show that a Hintikka set can be found along the path: Let U be the union of all formulae occurring in the left-hand side of each sequent along that path, and V be the union of all formulae occurring in the right-hand side of any such sequent. Let $S = \{\text{TA} \mid A \in U\} \cup \{\text{FA} \mid A \in V\} \cup \{\text{T}(t \approx t) \mid t \in H\}$. In order to show that

S is the Hintikka set with respect to H , we prove that the conditions $H0$ - $H8$ of Definition 30 hold for S .

$H0$ holds. Every atomic formula occurring in a sequent occurs in every path having this sequent as the source. Therefore, if S contains both TA and FA for some atom A , then some sequent in the path is an axiom. This contradicts the fact that the path is infinite. Also, for all $t \in H$, $F(t \approx t)$ is not in S for the same reason.

$H1$ - $H7$ hold. This can be shown in the same way as the proof of Lemma 9.

$H8$ holds. The case $H8(a)$ follows from the definition of S . As for $H8(b)$, it follows from the fact that all formulae in the lists $\mathcal{E}Q_0$ and $\mathcal{E}Q_i$ for all activated variables y_i , $i \geq 1$ are eventually entered in each infinite path of the tree.

Thus, S is a Hintikka set. By Lemma 18, some assignment $\sigma^{\mathfrak{H}_S}$ satisfies S in a structure \mathfrak{H}_S with the domain that is a quotient of H , and, thus, $\Gamma_0 \rightarrow \Delta_0$ is falsified by \mathfrak{H}_S and $\sigma^{\mathfrak{H}_S}$. \square

Lemma 20. *If the input sequent $\Gamma_0 \rightarrow \Delta_0$ is valid, then the procedure $Search_{\approx}$ halts with a closed tree T , from which a proof tree for a finite subsequent*

$$C_1, \dots, C_m \rightarrow D_1, \dots, D_n$$

of $\Gamma_0 \rightarrow \Delta_0$ can be constructed.

Proof. Similar to the proof of Lemma 10, but uses Lemma 19 instead of Lemma 9. \square

From Lemma 20 and Theorem 7 we get the following version of Gödel's extended completeness theorem for the calculus \mathbf{G}^{\approx} :

Theorem 8 (Completeness of \mathbf{G}^{\approx}). *A sequent (even infinite) is valid iff it is provable in \mathbf{G}^{\approx} .*

The following classical results apply also to languages with sequence variables, sequence functions and the equality predicate.

Theorem 9 (Löwenheim-Skolem, for \mathcal{L}_{\approx}). *If a set of formulae Γ of the language \mathcal{L}_{\approx} is satisfiable in some structure, then it is satisfiable in a structure whose domain is at most countably infinite.*

Theorem 10 (Compactness, for \mathcal{L}_{\approx}). *For any (possibly countably infinite) set Γ of formulae of the language \mathcal{L}_{\approx} , if every non-empty finite subset of Γ is satisfiable then Γ is satisfiable.*

Theorem 11 (Model Existence Theorem, for \mathcal{L}_{\approx}). *If a set of formulae Γ of the language \mathcal{L}_{\approx} is consistent then it is satisfiable.*

Lemma 21 (Consistency Lemma, for \mathcal{L}_{\approx}). *If a set of formulae Γ of the language \mathcal{L}_{\approx} is satisfiable then it is consistent.*

11 Advantages and Disadvantages of the Calculi \mathbf{G} and \mathbf{G}^\approx

The calculi \mathbf{G} and \mathbf{G}^\approx have many nice proof-theoretic properties, but they suffer drawbacks as well. First, they are not well suited for implementation because of too high non-determinism, and second, they do not completely reflect the intuitive meaning of sequence variables or sequence functions.

High non-determinism is a well-known problem for many sequent-based calculi (like, for instance, for classical \mathbf{LK}^\approx calculus). Degtyarev and Voronkov [17] survey methods to overcome it. In our case, a variant of basic superposition with ordering and equality constraint inheritance proposed in [30, 31] seems to be a reasonable alternative of \mathbf{G}^\approx , taking into account the fact that unification with sequence variables and sequence functions is infinitary but decidable [27]. This approach for theories with sequence variables, but without sequence functions has already been considered in [26]. To do the same for theories with sequence variables and sequence functions one needs to introduce a reduction ordering on terms involving sequence variables and functions, and an efficient algorithm for solving ordering constraints. This is a subject of further research and lays beyond the scope of this paper.

As for the second problem, it is natural to expect that a sequence term represents a finite sequence of individual terms. In other words, the intuition would suggest that the sequence terms should be interpreted as finite sequences of individual elements of the domain. However, it is not reflected in the theories we considered so far. We allow sequence terms to be interpreted with finite sequences that contain not only individual but also sequence elements of the domain. Therefore, many results one intuitively would expect to be proved do not hold. The example below illustrates one of such cases.

Example 3. Let tuple concatenation be defined by the following two equalities:

$$\forall \bar{x} \langle \rangle \asymp \langle \bar{x} \rangle \approx \langle \bar{x} \rangle. \quad (1)$$

$$\forall x \forall \bar{x} \forall \bar{y} \langle x, \bar{x} \rangle \asymp \langle \bar{y} \rangle \approx \langle x, \bar{x}, \bar{y} \rangle. \quad (2)$$

One might expect that under these assumptions the empty tuple plays the role of right neutral element for concatenation:

$$\forall \bar{x} \langle \bar{x} \rangle \asymp \langle \rangle \approx \langle \bar{x} \rangle. \quad (3)$$

However, \mathbf{G}^\approx does not prove it. If we have a closer look, this is justified, since (3) is not a logical consequence of (1) and (2): Just consider a structure $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$, where $\mathcal{D}_{\text{Ind}} = \{a, b, c\}$, $\mathcal{D}_{\text{Seq}} = \{d\}$, and \mathcal{I} is defined as follows:

– $\langle \rangle_{\mathcal{I}} : \mathcal{D}^\infty \rightarrow \mathcal{D}_{\text{Ind}}$, such that for all $n \geq 0$

$$\langle d_1, \dots, d_n \rangle_{\mathcal{I}} = \begin{cases} c, & \text{if } n = 0, \\ b, & \text{if } d_1 = d, \\ a, & \text{otherwise.} \end{cases}$$

– $\approx_{\mathcal{I}}: \mathcal{D}_{\text{Ind}}^2 \rightarrow \mathcal{D}_{\text{Ind}}$, defined as follows

$\approx_{\mathcal{I}}$	a	b	c
a	a	a	a
b	b	b	c
c	a	b	c

Let $\sigma^{\mathfrak{S}}$ be a state over \mathfrak{S} such that $\sigma^{\mathfrak{S}}(\bar{x}) = d$. Then the formulae (1) and (2) are true, but the formula (3) is false in \mathfrak{S} with respect to $\sigma^{\mathfrak{S}}$. It shows that (3) is not a logical consequence of (1) and (2).

If concatenation is defined as

$$\forall \bar{x} \forall \bar{y} \langle \bar{x} \rangle \approx \langle \bar{y} \rangle \approx \langle \bar{x}, \bar{y} \rangle \quad (4)$$

then (3) follows from (4) and \mathcal{G}_{\approx} proves it in two steps. It can be easily observed that these two ways of defining concatenation are equivalent in structures with the domain $\mathcal{D} = \mathcal{D}_{\text{Ind}}$.

To reflect the intended meaning of sequence terms in the semantics, below we consider structures with the domain $\mathcal{D} = \mathcal{D}_{\text{Ind}}$, i.e., with $\mathcal{D}_{\text{Seq}} = \emptyset$.

12 Induction with Sequence Variables

We start with the definitions of *inductive domain* and *intended structure*.

Definition 34. *let $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ be a structure for the language \mathcal{L}_{\approx} such that $\mathcal{D}_{\text{Seq}} = \emptyset$. Then \mathfrak{S} is called an intended structure for \mathcal{L}_{\approx} and \mathcal{D} is called an inductive domain.*

Note that Herbrand structures are *not* intended structures for \mathcal{L}_{\approx} . We will write $A[v]$ to indicate that the formula A contains a free occurrence of v .

Below we will use a special flexible arity function symbol f that satisfies the following axioms:

$$\begin{aligned} & \forall x \forall \bar{x} \forall \bar{y} \neg (f(\bar{x}, x, \bar{y}) \approx f()). \\ & \forall x \forall y \forall \bar{x} \forall \bar{y} f(x, \bar{x}) \approx f(y, \bar{y}) \Leftrightarrow x \approx y \wedge f(\bar{x}) \approx f(\bar{y}). \\ & \forall x \forall y \forall \bar{x} \forall \bar{y} f(\bar{x}, x) \approx f(\bar{y}, y) \Leftrightarrow f(\bar{x}) \approx f(\bar{y}) \wedge x \approx y. \\ & \forall \bar{x} \forall \bar{y} (f(\bar{x}) \approx f(\bar{y}) \wedge A\{\bar{z} \mapsto \bar{x}\}) \Rightarrow A\{\bar{z} \mapsto \bar{y}\}). \end{aligned}$$

where A is an arbitrary formula.

Definition 35. *The well-founded³ induction principle for sequence variables is formulated as follows:*

$$\forall \bar{x} (\forall \bar{y} (f(\bar{x}) \succ f(\bar{y}) \Rightarrow A\{\bar{x} \mapsto \bar{y}\}) \Rightarrow A[\bar{x}]) \Rightarrow \forall \bar{x} A[\bar{x}] \quad (\text{WFI})$$

where \succ is a well-founded ordering defined for terms with the head f .

³ Also called Noetherian.

Theorem 12. *Well-founded induction principle is valid.*

Proof. Let \mathfrak{S} be $\langle \mathcal{D}, \mathcal{I} \rangle$ and $\sigma^\mathfrak{S}$ be a state. Assume by contradiction that WFI is false in \mathfrak{S} with respect to $\sigma^\mathfrak{S}$. Then

$$\begin{aligned} \mathcal{V}al_\sigma^\mathfrak{S}(\forall \bar{x} (\forall \bar{y} f(\bar{x}) \succ f(\bar{y}) \Rightarrow A\{\bar{x} \mapsto \bar{y}\}) \Rightarrow A[\bar{x}]) &= \mathbf{T}, \\ \mathcal{V}al_\sigma^\mathfrak{S}(\forall \bar{x} A[\bar{x}]) &= \mathbf{F}. \end{aligned}$$

Then there exist $d_1, \dots, d_n \in \mathcal{D}$ such that

$$\mathcal{V}al_\sigma^\mathfrak{S}(A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) = \mathbf{F}.$$

Moreover, since \succ is a well-founded ordering for terms with the head f , we can assume without loss of generality that for all $c_1, \dots, c_m \in \mathcal{D}$,

$$\mathcal{V}al_\sigma^\mathfrak{S}(f(d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}}) \succ f(c_{1\mathfrak{S}}, \dots, c_{m\mathfrak{S}})) = \mathbf{T}$$

implies

$$\mathcal{V}al_\sigma^\mathfrak{S}(A\{\bar{x} \mapsto \ulcorner c_{1\mathfrak{S}}, \dots, c_{m\mathfrak{S}} \urcorner\}) = \mathbf{T}.$$

But then we have

$$\begin{aligned} \mathcal{V}al_\sigma^\mathfrak{S}((\forall \bar{y} f(d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}}) \succ f(\bar{y}) \Rightarrow A\{\bar{x} \mapsto \bar{y}\}) \Rightarrow \\ A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) &= \mathbf{F}, \end{aligned}$$

which implies that

$$\mathcal{V}al_\sigma^\mathfrak{S}(\forall \bar{x} (\forall \bar{y} f(\bar{x}) \succ f(\bar{y}) \Rightarrow A\{\bar{x} \mapsto \bar{y}\}) \Rightarrow A[\bar{x}]) = \mathbf{F},$$

a contradiction. □

Well-foundedness is an undecidable property. Therefore, we will develop syntactic instances of the WFI principle that avoid direct reference to arbitrary well-founded relations. We show some examples of such instantiation.

We start from auxiliary notions.

Definition 36. *The case distinction rule from the left is the formula*

$$(A\{\bar{x} \mapsto \ulcorner \urcorner\} \wedge \forall y \forall \bar{y} A\{\bar{x} \mapsto \ulcorner y, \bar{y} \urcorner\}) \Rightarrow \forall \bar{x} A[\bar{x}] \quad (\text{LCD})$$

The LCD rule is not valid, as the following example shows:

Example 4. Let $A[\bar{x}]$ in LCD be the atom $p(\bar{x})$. Take a structure $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ such that $\mathcal{D}_{\text{Ind}} = \{a\}$, $\mathcal{D}_{\text{Seq}} = \{b\}$ and $p_{\mathcal{I}}$ contains all the finite tuples over \mathcal{D} whose first element is not b . Then LCD is false in \mathfrak{S} .

However, the following theorem holds:

Theorem 13. *LCD is true in every intended model.*

Proof. Let $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ be an intended structure and $\sigma^\mathfrak{S}$ be a state over \mathfrak{S} . Assume LCD is false in \mathfrak{S} with respect to $\sigma^\mathfrak{S}$. Then we have

$$\begin{aligned} \mathcal{V}al_\sigma^\mathfrak{S}(A\{\bar{x} \mapsto \ulcorner \urcorner\} \wedge \forall y \forall \bar{y} A\{\bar{x} \mapsto \ulcorner y, \bar{y} \urcorner\}) &= \mathbf{T}, \\ \mathcal{V}al_\sigma^\mathfrak{S}(\forall \bar{x} A[\bar{x}]) &= \mathbf{F}. \end{aligned}$$

Then there exist $d_1, \dots, d_n \in \mathcal{D} = \mathcal{D}_{\text{Ind}}$ such that

$$\mathcal{V}al_\sigma^\mathfrak{S}(A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) = \mathbf{F}.$$

If $n = 0$, we get $\mathcal{V}al_\sigma^\mathfrak{S}(A\{\bar{x} \mapsto \ulcorner \urcorner\}) = \mathbf{F}$, which contradicts the assumption. If $n > 0$, since $d_{1\mathfrak{S}}$ is an individual term, we get $\forall y \forall \bar{y} A\{\bar{x} \mapsto \ulcorner y, \bar{y} \urcorner\}) = \mathbf{F}$, again a contradiction. \square

Definition 37. The suffix ordering \succ_{Suf} is defined on terms with the head f :

$$\begin{aligned} \forall \bar{x} \forall \bar{y} (f(\bar{x}) \succ_{\text{Suf}} f(\bar{y})) &\Leftrightarrow \\ \exists z \exists \bar{z} f(\bar{x}) \approx f(z, \bar{z}) \wedge (f(\bar{z}) \approx f(\bar{y}) \vee f(\bar{z}) \succ_{\text{Suf}} f(\bar{y})) &\quad (\text{SO}) \end{aligned}$$

Suffix ordering is well-founded and has the property that any term of the form $f(t_1, \dots, t_n)$, $n > 0$, which is not minimal with respect to \succ_{Suf} , has individual terms as its first k arguments for some $1 \leq k \leq n$.

Definition 38. The structural induction from the left is the formula

$$(A\{\bar{x} \mapsto \ulcorner \urcorner\} \wedge \forall y \forall \bar{y} (A\{\bar{x} \mapsto \bar{y}\} \Rightarrow A\{\bar{x} \mapsto \ulcorner y, \bar{y} \urcorner\})) \Rightarrow \forall \bar{x} A[\bar{x}]. \quad (\text{LSI})$$

LSI can be obtained syntactically from SO and the instances of WFI and LCD: If we take

$$\begin{aligned} B[\bar{x}] &:= \forall \bar{y} (f(\bar{x}) \succ_{\text{Suf}} f(\bar{y}) \Rightarrow A\{\bar{x} \mapsto \bar{y}\}) \Rightarrow A[\bar{x}], \\ \text{WFI}' &:= \forall \bar{x} B[\bar{x}] \Rightarrow \forall \bar{x} A[\bar{x}], \\ \text{LCD}' &:= (B\{\bar{x} \mapsto \ulcorner \urcorner\} \wedge \forall z \forall \bar{z} B\{\bar{x} \mapsto \ulcorner z, \bar{z} \urcorner\}) \Rightarrow \forall \bar{x} B[\bar{x}], \end{aligned}$$

then the following theorem holds:

Theorem 14. The sequent $\text{SO}, \text{LCD}', \text{WFI}' \rightarrow \text{LSI}$ is provable in \mathbf{G}^\approx .

We proved this theorem with the help of one of the provers of the THEOREMA system. First, we proved LSI from SO, LCD', and WFI' automatically by the THEOREMA prover for predicate logic using metavariables, called PLM [24], and then translated the output into the \mathbf{G}^\approx proof. (The calculus implemented in PLM is different from \mathbf{G}^\approx : it uses metavariables and has a restricted (incomplete) sequence unification algorithm for sequence metavariables.)

Theorem 15. LSI is true in every intended model.

Proof. Let $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ be an intended structure and $\sigma^\mathfrak{S}$ be a state over \mathfrak{S} . Assume LCD is false in \mathfrak{S} with respect to $\sigma^\mathfrak{S}$. Then we have

$$\begin{aligned} \mathcal{V}al_\sigma^\mathfrak{S}(A\{\bar{x} \mapsto \ulcorner \urcorner\} \wedge \forall y \forall \bar{y} (A\{\bar{x} \mapsto \bar{y}\} \Rightarrow A\{\bar{x} \mapsto \ulcorner y, \bar{y} \urcorner\})) &= \mathbf{T}, \\ \mathcal{V}al_\sigma^\mathfrak{S}(\forall \bar{x} A[\bar{x}]) &= \mathbf{F}. \end{aligned}$$

Then there exist $d_1, \dots, d_n \in \mathcal{D} = \mathcal{D}_{\text{Ind}}$ such that

$$\mathcal{V}al_\sigma^\mathfrak{S}(A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\}) = \mathbf{F}$$

and for all $m < n$ and for all $e_1, \dots, e_m \in \mathcal{D}$

$$\mathcal{V}al_\sigma^\mathfrak{S}(A\{\bar{x} \mapsto \ulcorner e_{1\mathfrak{S}}, \dots, e_{m\mathfrak{S}} \urcorner\}) = \mathbf{T}.$$

If $n = 0$, we get $\mathcal{V}al_\sigma^\mathfrak{S}(A\{\bar{x} \mapsto \ulcorner \urcorner\}) = \mathbf{F}$, which contradicts the assumption. If $n > 0$, then $A\{\bar{x} \mapsto \ulcorner d_{2\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\} \Rightarrow A\{\bar{x} \mapsto \ulcorner d_{1\mathfrak{S}}, \dots, d_{n\mathfrak{S}} \urcorner\} = \mathbf{F}$ and, hence, $\forall y \forall \bar{y} (A\{\bar{x} \mapsto \bar{y}\} \Rightarrow A\{\bar{x} \mapsto \ulcorner y, \bar{y} \urcorner\}) = \mathbf{F}$, again a contradiction. \square

The next theorem, in fact, shows that LCD can be proved from LSI:

Theorem 16. *The sequent $\rightarrow \text{LSI} \Rightarrow \text{LCD}$ is provable in \mathbf{G}^\approx .*

Like Theorem 14, we proved this theorem automatically using the THEOREMA system. Now we turn LSI into the inference rule:

$$\frac{\Gamma \rightarrow \Delta, A\{\bar{x} \mapsto \ulcorner \urcorner\} \wedge \forall y \forall \bar{y} (A\{\bar{x} \mapsto \bar{y}\} \Rightarrow A\{\bar{x} \mapsto \ulcorner y, \bar{y} \urcorner\}), A}{\Gamma \rightarrow \Delta, \forall \bar{x} A[\bar{x}], A} \quad (\text{SI}_{\text{left}})$$

Soundness of \mathbf{G}^\approx and Theorem 15 imply the following result:

Theorem 17. *If a sequent is provable using $(\text{SI}_{\text{left}})$ and the inference rules of \mathbf{G}^\approx , then it is true in every intended structure.*

In the similar way we can get another instance of WFI.

Definition 39. *The case distinction rule from the right is the formula*

$$(A\{\bar{x} \mapsto \ulcorner \urcorner\} \wedge \forall \bar{y} \forall y A\{\bar{x} \mapsto \ulcorner \bar{y}, y \urcorner\}) \Rightarrow \forall \bar{x} A[\bar{x}] \quad (\text{RCD})$$

The prefix ordering \succ_{Pre} is defined on terms with the head f as follows:

$$\begin{aligned} \forall \bar{x} \forall \bar{y} (f(\bar{x}) \succ_{\text{Pre}} f(\bar{y})) &\Leftrightarrow \\ \exists \bar{z} \exists z f(\bar{x}) \approx f(\bar{z}, z) \wedge (f(\bar{z}) \approx f(\bar{y}) \vee f(\bar{z}) \succ_{\text{Pre}} f(\bar{y})) &\quad (\text{PO}) \end{aligned}$$

The structural induction from the right is the formula

$$(A\{\bar{x} \mapsto \ulcorner \urcorner\} \wedge \forall \bar{y} \forall y (A\{\bar{x} \mapsto \bar{y}\} \Rightarrow A\{\bar{x} \mapsto \ulcorner \bar{y}, y \urcorner\})) \Rightarrow \forall \bar{x} A[\bar{x}]. \quad (\text{RSI})$$

By the same reasoning as for LSI above, we can turn RSI into an inference rule:

$$\frac{\Gamma \rightarrow \Delta, A\{\bar{x} \mapsto \ulcorner \urcorner\} \wedge \forall \bar{y} \forall y (A\{\bar{x} \mapsto \bar{y}\} \Rightarrow A\{\bar{x} \mapsto \ulcorner \bar{y}, y \urcorner\}), A}{\Gamma \rightarrow \Delta, \forall \bar{x} A[\bar{x}], A} \quad (\text{SI}_{\text{right}})$$

The counterpart of Theorem 17 hold for $(\text{SI}_{\text{right}})$.

Another useful well-founded ordering on terms with sequence variables is the *length ordering* defined as follows:

$$\begin{aligned} \forall \bar{x} \forall \bar{y} (f(\bar{x}) \succ_{\text{Len}} f(\bar{y}) &\Leftrightarrow \exists z \exists \bar{z} f(\bar{x}) \approx f(z, \bar{z}) \\ \wedge (f(\bar{y}) \approx f() \vee \exists u \exists \bar{u} (f(\bar{y}) &\approx f(u, \bar{u}) \wedge f(\bar{z}) \succ_{\text{Len}} f(\bar{u}))) \end{aligned} \quad (\text{LO})$$

If we instantiate the ordering \succ in WFI with \succ_{Len} , we get an instance of WFI called *well-founded induction principle with the length ordering*:

$$\forall \bar{x} (\forall \bar{y} (f(\bar{x}) \succ_{\text{Len}} f(\bar{y}) \Rightarrow A\{\bar{x} \mapsto \bar{y}\}) \Rightarrow A[\bar{x}]) \Rightarrow \forall \bar{x} A[\bar{x}] \quad (\text{WFILO})$$

The next example shows that WFILO is not valid.

Example 5. Let $A[\bar{x}]$ be an atom $p(\bar{x})$ with the flexible arity predicate symbol p . Let $\mathfrak{S} = \langle \mathcal{D}, \mathcal{I} \rangle$ be a structure whose domain \mathcal{D} is the set of all ground terms built from an individual constant c , sequence constant \bar{c} and a flexible arity function symbol f . The assignment \mathcal{I} is defined so that it maps each ground term to itself and the predicate \succ_{Len} to the same predicate on \mathcal{D} . As for $p_{\mathcal{I}}$, let it be a flexible arity predicate on \mathcal{D} that contains all the tuples over \mathcal{D} except $\langle c, \bar{c} \rangle$. Then $\text{Val}_{\sigma}^{\mathfrak{S}}(f(c, \bar{c}) \succ_{\text{Len}} f() \Rightarrow p() \Rightarrow p(c, \bar{c})) = \mathbf{F}$ which implies that $\text{Val}_{\sigma}^{\mathfrak{S}}(\forall \bar{x} (\forall \bar{y} f(\bar{x}) \succ_{\text{Len}} f(\bar{y}) \Rightarrow p(\bar{y})) \Rightarrow p(\bar{x})) = \mathbf{T}$. On the other side, $\text{Val}_{\sigma}^{\mathfrak{S}}(\forall \bar{x} p(\bar{x})) = \mathbf{F}$. Therefore, WFILO is false in \mathfrak{S} with respect to $\sigma^{\mathfrak{S}}$.

Nevertheless, WFILO is satisfied by any intended structure:

Theorem 18. *WFILO is true in any intended structure.*

Proof. Using the same argument as in the proof of the Theorem 12, but on inductive domains. \square

Instantiating f in WFILO with the tuple constructor we get the tuple induction principle formulated in [11].

WFILO can be turned into an inference rule.

Definition 40. *The inference rule of well-founded induction with the length ordering is defined as*

$$\frac{\Gamma, \text{LO} \rightarrow \Delta, \forall \bar{x} (\forall \bar{y} (f(\bar{x}) \succ_{\text{Len}} f(\bar{y}) \Rightarrow A\{\bar{x} \mapsto \bar{y}\}) \Rightarrow A[\bar{x}]), A}{\Gamma, \text{LO} \rightarrow \Delta, \forall \bar{x} A[\bar{x}], A} \quad (\text{WFI}_{\text{len}})$$

The counterpart of Theorem 17 holds for $(\text{WFI}_{\text{len}})$.

The calculus \mathbf{G}^{\approx} does not contain the cut rule, but we need it in induction proofs.

$$\frac{\Gamma \rightarrow \Delta, A, A \quad \Gamma, A \rightarrow \Delta, B, A}{\Gamma \rightarrow \Delta, B, A} \quad (\text{Cut})$$

Cut rule forms a basis for THEOREMA conjecture generation algorithm and the cascade method introduced by the first author in [9].

13 Encoding in Order-Sorted Logic

There is a waste literature on order-sorted logic and its applications in automated reasoning and knowledge representation (see, for instance, [35, 34, 5, 14, 23, 36]). In this section we show that predicate logic with sequence variables and sequence functions can be encoded as a special order-sorted first-order theory.

The set of sorts consists of two elements **ind** and **seq** ordered as $\text{ind} \leq \text{seq}$. Individual variables have sort **ind** and sequence variables have sort **seq**. Function declarations are defined as follows: If $f \in \mathcal{F}_{\text{Ind}}^{\text{Fix}}$ with $\mathcal{A}r(f) = n$, then we have $f : \underbrace{\text{ind} \times \cdots \times \text{ind}}_{n \text{ times}} \rightarrow \text{ind}$; If $f \in \mathcal{F}_{\text{Ind}}^{\text{Flex}}$, then $f : \text{seq} \rightarrow \text{ind}$; If $f \in \mathcal{F}_{\text{Seq}}^{\text{Fix}}$ with $\mathcal{A}r(f) = n$, then $f : \underbrace{\text{ind} \times \cdots \times \text{ind}}_{n \text{ times}} \rightarrow \text{seq}$; If $f \in \mathcal{F}_{\text{Seq}}^{\text{Flex}}$, then $f : \text{seq} \rightarrow \text{seq}$. We have two special functions: binary $\ulcorner \urcorner$ and constant $\llbracket \rrbracket$ with the declarations $\ulcorner \urcorner : \text{seq} \times \text{seq} \rightarrow \text{seq}$ and $\llbracket \rrbracket : \text{seq}$. Predicate declarations consist of $p : \underbrace{\text{ind} \times \cdots \times \text{ind}}_{n \text{ times}}$ for each $p \in \mathcal{P}^{\text{Fix}}$ with $\mathcal{A}r(p) = n$, and $p : \text{seq}$ for each $p \in \mathcal{P}^{\text{Flex}}$. In this way, flexible arity symbols are encoded as unary ones. As for the inference system, we can use any calculus for order-sorted first-order logic augmented with the following three rules (axioms) for $\ulcorner \urcorner$ and $\llbracket \rrbracket$: associativity $\ulcorner s, \ulcorner t, r \urcorner \urcorner \approx_{\text{seq}} \ulcorner \ulcorner s, t \urcorner, r \urcorner$, right identity $\ulcorner s, \llbracket \rrbracket \urcorner \approx_{\text{seq}} s$, left identity $\ulcorner \llbracket \rrbracket, s \urcorner \approx_{\text{seq}} s$. The results of this paper prove completeness of such an extended calculus.

14 Conclusion

We described a syntax, semantics and inference system for a logic with sequence variables and sequence functions. The calculus \mathbf{G}^{\approx} for such a logic extends Gentzen's \mathbf{LK}^{\approx} calculus and has many nice proof-theoretic properties. On the other side, in order to fully reflect the intuition behind sequence variables (abbreviation of finite sequences of individual terms), we introduced induction with sequence variables and defined several versions of induction rules. The interesting feature of logic with sequence variables and sequence functions is that there is no need in introducing constructors to define inductive data types. This information is “built-in” into the logic itself. We showed how to encode the logic with sequence variables and sequence function symbols as a special order-sorted first-order theory.

References

1. An Interactive Mathematical Proof System. <http://imps.mcmaster.ca/>.
2. The COQ Proof Assistant. <http://coq.inria.fr/>.
3. The MIZAR Project. <http://www.mizar.org/>.
4. The OMEGA System. <http://www.ags.uni-sb.de/~omega/>.
5. C. Beierle, U. Hedtstück, U. Pletat, P H. Schmitt, and J. Siekmann. An order-sorted logic for knowledge representation systems. *Artificial Intelligence*, 55:149–191, 1992.

6. W. W. Bledsoe and Guohui Feng. SET-VAR. *J. Automated Reasoning*, 11(3):293–314, 1993.
7. H. Boley. *A Tight, Practical Integration of Relations and Functions*, volume 1712 of *LNAI*. Springer Verlag, 1999.
8. B. Buchberger. Mathematica as a rewrite language. In T. Ida, A. Ohori, and M. Takeichi, editors, *Proceedings of the 2nd Fuji International Workshop on Functional and Logic Programming*, pages 1–13, Shonan Village Center, Japan, 1–4 November 1996. World Scientific.
9. B. Buchberger. Theory exploration with THEOREMA. *Analele Universitatii Din Timisoara, ser. Matematica-Informatica*, XXXVIII(2):9–32, 2000.
10. B. Buchberger. Algorithm invention and verification by lazy thinking. In D. Petcu, D. Zaharie, V. Negru, and T. Jebelean, editors, *Proc. of the 5rd Int. Workshop on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'03)*, pages 2–26, Timisoara, Romania, 1–4 October 2003. Mirton.
11. B. Buchberger and A. Craciun. Algorithm synthesis by lazy thinking: Examples and implementation in THEOREMA. In *Proc. of the Mathematical Knowledge Management Symposium*, volume 93 of *ENTCS*, pages 24–59, Edunburgh, UK, 25–29 November 2003. Elsevier Science.
12. B. Buchberger, C. Dupré, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru, and W. Windsteiger. The Theorema project: A progress report. In M. Kerber and M. Kohlhase, editors, *Symbolic Computation and Automated Reasoning. Proceedings of Calculemus'2000 Conference*, pages 98–113, St. Andrews, UK, 6–7 August 2000.
13. E. Clarke, M. Kohlhase, J. Ouaknine, and K. Sutner. System description: ANALYTICA 2. In T. Hardin and R. Rioboo, editors, *Proc. of the 11th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (Calculemus'03)*, Rome, Italy, 10–12 September 2003. Aracne Editrice S.R.L.
14. A. G. Cohn. A many sorted logic with possibly empty sorts. In D. Kapur, editor, *Proc. of the 11th Int. Conference on Automated Deduction, CADE'92*, volume 607 of *LNAI*, pages 633–647, Saratoga Springs, US, 15–18 June 1992. Springer Verlag.
15. R. Constable. *Implementing Mathematics Using the NUPRL Proof Development System*. Prentice-Hall, 1986.
16. N. G. de Bruijn. The mathematical language AUTOMATH, its usage, and some of its extensions. In M. Laudet, D. Lacombe, L. Nolin, and M. Schützenberger, editors, *Proc. of Symposium on Automatic Demonstration, Versailles, France*, volume 125 of *LN in Mathematics*, pages 29–61. Springer Verlag, Berlin, 1970.
17. A. Degtyarev and A. Voronkov. Equality reasoning in sequent-based calculi. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 10, pages 613–706. Elsevier Science, 2001.
18. M. R. Genesereth, Ch. Petrie, T. Hinrichs, A. Hondroulis, M. Kassoff, N. Love, and W. Mohsin. Knowledge Interchange Format, draft proposed American National Standard (dpANS). Technical Report NCITS.T2/98-004, 1998. <http://logic.stanford.edu/kif/dpans.html>.
19. G. Gentzen. Untersuchungen über das logische schließen. *Mathematische Zeitschrift*, 39:176–210, 1934.
20. M. Gordon and T. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher-Order Logic*. Cambridge University Press, 1993.
21. Common Logic Working Group. Common logic: Abstract syntax and semantics. <http://cl.tamu.edu/docs/cl/1.0/cl-1.0.pdf>, 2003.
22. P. Hayes and C. Menzel. Semantics of knowledge interchange format. <http://reliant.tekknowledge.com/IJCAI01/HayesMenzel-SKIF-IJCAI2001.pdf>, 2001.

23. M. Kohlhase. A mechanization of sorted higher-order logic based on the resolution principle. PhD Thesis. Universität des Saarlandes. Saarbrücken, Germany, 1994.
24. B. Konev and T. Jebelean. Using meta-variables for natural deduction in THEOREMA. In M. Kerber and M. Kohlhase, editors, *Proc. of Calculemus'2000 Conference*, St. Andrews, UK, 6–7 August 2000.
25. T. Kutsia. Solving and proving in equational theories with sequence variables and flexible arity symbols. Technical Report 02-31, RISC, Johannes Kepler University, Linz, Austria, 2002.
26. T. Kutsia. Theorem proving with sequence variables and flexible arity symbols. In M. Baaz and A. Voronkov, editors, *Logic in Programming, Artificial Intelligence and Reasoning. Proceedings of the 9th International Conference LPAR'02*, volume 2514 of *Lecture Notes in Artificial Intelligence*, pages 278–291, Tbilisi, Georgia, 14–18 September 2002. Springer Verlag.
27. T. Kutsia. Solving equations involving sequence variables and sequence functions. Technical Report 04-01, Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria, 2004.
28. Zh. Luo and R. Pollack. LEGO proof development system: User's manual. Technical Report ECS-LFCS-92-211, University of Edinburgh, 1992.
29. L. Magnusson and B. Nordström. The ALF proof editor and its proof engine. In H. Barendregt and T. Nipkow, editors, *Types for Proofs and Programs*, volume 806 of *LNCS*, pages 213–237. Springer Verlag, 1994.
30. R. Nieuwenhuis and A. Rubio. Theorem proving with ordering constrained clauses. In D. Kapur, editor, *Proceedings of the 11th International Conference on Automated Deduction, CADE'92*, volume 814 of *Lecture Notes in Artificial Intelligence*, Saratoga Springs, New York, US, 15–18 June 1992. Springer Verlag.
31. R. Nieuwenhuis and A. Rubio. Theorem proving with ordering and equality constrained clauses. *Journal of Symbolic Computation*, 19:321–351, 1995.
32. L. Paulson. ISABELLE: the next 700 theorem provers. In P. Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press, 1990.
33. F. Pfenning. ELF: A meta-language for deductive systems. In A. Bundy, editor, *Proc. of the 12th International Conference on Automated Deduction, CADE'94*, volume 814 of *LNAI*, pages 811–815, Nancy, France, 1995. Springer Verlag.
34. M. Schmidt-Schauss. *Computational aspects of an order-sorted logic with term declarations*, volume 395 of *LNAI*. Springer Verlag, 1989.
35. C. Walther. *A many sorted calculus based on resolution and paramodulation*. Pitman, London, 1987.
36. C. Weidenbach. Computational aspects of first-order logic with sorts. PhD Thesis. Universität des Saarlandes. Saarbrücken, Germany, 1996.
37. W. Windsteiger. Exploring an algorithm for polynomial interpolation in the THEOREMA system. In T. Hardin and R. Rioboo, editors, *Proc. of the 11th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (Calculemus'03)*, pages 130–136, Rome, Italy, 10–12 September 2003. Aracne Editrice S.R.L.
38. S. Wolfram. *The Mathematica Book*. Cambridge University Press and Wolfram Research, Inc., fourth edition, 1999.

Index

- $\mathcal{A}vail_i$, 21
- \mathbf{G} , 13
- \mathbf{G}^\approx , 30
- $\mathcal{T}(\mathcal{F}, \mathcal{V})$, 4
- $\mathcal{T}_{\text{Ind}}(\mathcal{F}, \mathcal{V})$, 4
- $\mathcal{T}_{\text{Seq}}(\mathcal{F}, \mathcal{V})$, 4
- $\cong_{\mathcal{H}_{\text{Ind}}}^E$, 31
- $\cong_{\mathcal{H}}^E$, 32
- Dom , 6
- Ind_0 , 20
- Ind_0^{Av} , 20
- Ind_i^{Av} , 21
- Inds , 20
- \mathcal{V}^{New} , 19
- $\mathcal{V}_{\text{Ind}}^{\text{New}}$, 20
- $\mathcal{V}_{\text{Seq}}^{\text{New}}$, 20
- $\mathcal{V}^{\text{Used}}$, 19
- $\mathcal{V}_{\text{Ind}}^{\text{Used}}$, 20
- $\mathcal{V}_{\text{Seq}}^{\text{Used}}$, 20
- Act_{Ind} , 21
- Act_{Top} , 21
- Terms , 20
- Top_0 , 20
- Top_0^{Av} , 20
- Top_i^{Av} , 21
- Tuples , 20
- ε , 5
- substitute*, 23
- v*-variant, 7
- Expand**, 23
- Expand** $^\approx$, 37
- Grow-Left**, 24
- Grow-Right**, 25
- Search**, 22
- Search** $^\approx$, 36

- Antecedent, 13
- Atom, 5
- Atomic formula, 5
- Axiom
 - of \mathbf{G} , 14
 - of \mathbf{G}^\approx , 30
 - of reflexivity for \approx , 30
- Binding
 - of a variable, 5
- Case distinction
 - from the left, 41
 - from the right, 43
- Closed
 - leaf, 15
 - tree, 15
- Compactness theorem
 - for \mathcal{L} , 29
 - for \mathcal{L}^\approx , 38
- Completeness theorem
 - for \mathbf{G} , 29
 - for \mathbf{G}^\approx , 38
- Component of a signed formula, 11
- Conclusion, 14
 - of a deduction tree, 15
 - of a proof tree, 15
- Congruence, 31
- Conjugate, 12
- Consistency lemma
 - for \mathcal{L} , 29
 - for \mathcal{L}^\approx , 38

- Deduction tree, 15
- Domain
 - inductive, 40
 - of a structure, 6

- Eigenvariable, 14
- Expansion
 - of a language, 13
 - of a structure, 13
- Extended language, 8

- Finished leaf
 - in **Search**, 22
 - in **Search** $^\approx$, 35
- Formula
 - atomic, 5
 - principal, 14
 - satisfiable, 8
 - signed, 11
 - signed, of type α , 11
 - signed, of type β , 11
 - signed, of type δ , 11

- signed, of type γ , 11
- true is a structure, 8
- valid, 8
- Function symbol
 - individual, of fixed arity, 4
 - individual, of flexible arity, 4
 - sequence, of fixed arity, 4
 - sequence, of flexible arity, 4
- Herbrand universe, 18
- Hintikka set
 - for languages with equality, 31
 - for languages without equality, 17
- Individual function symbol
 - of fixed arity, 4
 - of flexible arity, 4
- Individual variable, 4
- Induction
 - structural, from the left, 42
- Inductive domain, 40
- Instance, 6
- Intended structure, 40
- Interpretation, 7
- Löwenheim-Skolem theorem
 - for \mathcal{L} , 29
 - for \mathcal{L}_{\approx} , 38
- Language, 4
 - extended, 8
 - with equality, 4
 - without equality, 4
- Leaf
 - closed, 15
 - finished, in **Search**, 22
 - finished, in **Search $_{\approx}$** , 35
- Lemma
 - of consistency, for \mathcal{L} , 29
 - of consistency, for \mathcal{L}_{\approx} , 38
- Logical consequence, 8
- Model, 8
- Model existence theorem
 - for \mathcal{L} , 29
 - for \mathcal{L}_{\approx} , 38
- Name, 8
- Non-logical symbols, 4
- Ordering
 - prefix, 43
- Predicate symbol
 - of fixed arity, 4
 - of flexible arity, 4
- Prefix ordering, 43
- Premise, 14
- Principal formula, 14
- Principle
 - of well-founded induction, 40
- Procedure
 - **Expand**, 23
 - **Expand $_{\approx}$** , 37
 - **Grow-Left**, 24
 - **Grow-Right**, 25
 - **Search**, 22
 - **Search $_{\approx}$** , 36
- Proof tree, 15
- Reduct
 - of a language, 13
 - of a language with respect to a set of signed formulae, 13
 - of a structure, 13
- Reflexivity axiom for \approx , 30
- Satisfiable formula, 8
- Sentence, 5
- Sequence function symbol
 - of fixed arity, 4
 - of flexible arity, 4
- Sequence variable, 4
- Sequent
 - falsifiable, 14
 - inconsistent, 13
 - provable, 15
 - valid, 15
- Sequent calculus
 - **G**, 13
 - **G $_{\approx}$** , 30
- Signed formula, 11
 - of type α , 11
 - of type β , 11
 - of type δ , 11
 - of type γ , 11
- Soundness theorem
 - for **G**, 17
 - for **G $_{\approx}$** , 30
- State, 7
- Structural induction

- from the left, 42
- Structure, 6
 - intended, 40
- Succedent, 13
- Term
 - free for a variable in a formula, 6
- term, 4
- Term algebra, 12
- Theorem
 - compactness, for \mathcal{L} , 29
 - compactness, for \mathcal{L}_{\approx} , 38
 - completeness, for \mathbf{G} , 29
 - completeness, for \mathbf{G}^{\approx} , 38
 - Löwenheim-Skolem, for \mathcal{L} , 29
 - Löwenheim-Skolem, for \mathcal{L}_{\approx} , 38
- of model existence, for \mathcal{L} , 29
- of model existence, for \mathcal{L}_{\approx} , 38
- soundness, for \mathbf{G} , 17
- soundness, for \mathbf{G}^{\approx} , 30
- Tree
 - closed, 15
- Truth value, 7
- Valid formula, 8
- Value, 7
- Variable
 - individual, 4
 - sequence, 4
- Variable binding, 5
- Well-founded induction principle, 40