A Proof Calculus with Case Analysis as its Only Rule

Ján Kľuka and Paul J. Voda

Dept. of Applied Informatics, Faculty of Mathematics, Physics, and Informatics,
Comenius University, Mlynská dolina, 842 48 Bratislava, Slovakia
{kluka,voda}@fmph.uniba.sk

1 Introduction

In [KV09] we have presented a proof calculus for first-order logic based on a single rule of *case analysis* (cut). The calculus was to be used in our intelligent proof assistant (IPA) system called CL. The proofs were binary trees labeled with sentences. The idea was that $D :\equiv {}^{A}\langle D_1, D_2\rangle$ proves a sequent $\Gamma \Rightarrow \Delta$ iff D_1 proves $\Gamma, A \Rightarrow \Delta$ and D_2 proves $\Gamma \Rightarrow \Delta, A$. Although the system looked like a Hilbert-style system (one rule, many axioms), it had a quite pleasing property of being able to simulate directly derivations in sequent calculi.

In the present self-contained paper we have made two substantial changes to this treatment. First, we have realized that proofs D can be read as *if-then-else* formulas used in some declarative programming languages with the meaning $(A \wedge D_1) \vee (\neg A \wedge D_2)$. Instead of turning formulas to proofs (as it is sometimes done in logic [Sch77,Ane90]), we have reversed the situation and constructed formulas as if they were case analysis proofs. We have a system with a single propositional proof rule:

> *a path in an* if-then-else *formula leading to its* $\bot$ *leaf is inconsistent with the formula.*

This is extended to the full quantification logic in a style somewhat reminiscent of Hilbert's ϵ-calculus. Instead of Skolem functions (a.k.a. ϵ-terms) we use Henkin witnessing constants.

The second change to [KV09] concerns *definition hiding*. Program verification involves a series of extensions of a language $\mathcal{L}$ by definitions of symbols $\vec{F}$, followed by the proof of a property A of $\vec{F}$. However, one would often prefer to prove A as a *specification* of $\vec{F}$, and hide the details of definitions within the proof. Our proof system is explicitly designed to make this possible. At the end of the paper, we will put forth an argument in favor of definition hiding in pure mathematical logic.

If-then-else constructs are also used in binary decision diagrams (BDD) to represent boolean functions. BDDs are intensively studied mainly in connection with the verification of VLSI circuits ([Bry86]). As the experience with BDD confirms, *if-then-else* formulas seem to capture the very essence of propositional logic, namely, that a formula is a tautology if *no consistent path through it leads*

to $\bot$. This intuition is more natural than the one achieved via the truth tables of the usual connectives. Paths through formulas play central role to our definition of provability.

When designing our calculus we have known about the role of the *if-then-else* connective in BDDs, but we have arrived at its usefulness by analyzing our previous attempt at a simple proof calculus. We have thought that BDDs were concerned only with propositional logic. Only when we were finishing this paper, we came across three instances of using BDDs as the basis for quantification logic [PS95,GT03,Gou94]. All three are intended for the automated theorem proving (ATP) community. Both the presentation and the means used (such as unification, the restrictions of formulas A in the above D to prime formulas) are specific to the community.

We think that we have succeeded in formulating our calculus in purely logical terms free from the computer science terminology and notation (with the sole exception of finite sequences in Par. 3.4, where we think that the well-tried out computer science notation is superior). This should make the power and elegance of the *if-then-else* connective available to the logic community at large. Besides, the couching of calculus in human terms is essential in the IPA approach, while it is less important in ATP (see Sect. 2). We consider this our first contribution. Our second contribution is the already mentioned extensibility with definition hiding.

Since not all computer scientists are expert logicians, we carry out the logical arguments in a more detail than usual. Since not all logicians are expert computer scientists, we discuss IPAs in Sect. 2 in a more detail than usual. Sections 3 and 4 deal with the syntax (formulas, proofs). In Sect. 5 we present a surprisingly simple and elegant proof of completeness of our calculus. In Sect. 6 we discuss the incremental construction of proofs by (local) derivation rules and show that the proofs can be presented both in Hilbert and Gentzen styles.

2 Intelligent Proof Assistants

This is a short overview of assisted theorem proving for logicians. Readers familiar with IPAs can skip over it.

One of the main motivations for the development of our proof calculus was entirely pragmatical. It was to be a formal basis for a new version of our programming language CL (Clausal Language) [CL97] which will come with a new IPA. Programs in CL are just certain implications in extensions by definitions of Peano Arithmetic. We have successfully used CL for the last ten years in the teaching of first-order logic and also in our courses on program verification.

IPAs (e.g., Isabelle [NPW02], Coq [BC04]) are distinguished from ATPs in that the systematic proof searches, so typical for the latter, are not crucial. Neither is an IPA prima facie an applied logical vehicle for the investigation of cut elimination and proof transformations, including extraction of witnesses. The case for an IPA can be briefly stated as follows:

- The software development is quite error prone and formal vehicles for program verification are sorely needed.
- Programs are constructed to accept inputs of arbitrary size, and thus the main formal verification vehicle must of necessity be induction. ATPs cannot generally find induction formulas automatically. Specialized SAT-solvers used in hardware design typically deal with fixed sizes (i.e., the number of logical gates).
- Quantifiers are necessary in the statement of program properties. Determining the witnessing terms is best left to humans. Unification used in some ATPs degenerates to enumeration in the worst case.
- The construction of programs – which amounts to definition of primitive recursive functions (or functionals) of rather low complexity (Grzegorczyk's $\mathcal{E}^4$ is already unfeasible) – is a difficult, mentally straining activity. The additional effort of verifying the programs usually more than doubles the level of difficulty of programming.
- For many, so called mission critical programs, the correctness is crucial as human lives and/or expensive equipment depend on their correct functioning.
- The correctness proofs are formally verified properties of concrete functions, it is essential that the proofs are developed to details which can be machine checked.
- The proofs are constructed by humans in that they formulate lemmas, choose induction formulas, and guess witnessing terms.
- Although most IPAs work in sequent or tableau calculi with subformula properties, the use of cuts (i.e., modus ponens) is crucial. This is because the properties must be proved in a natural Hilbert style, just as it is done in mathematics. This is to be contrasted with ATPs where the formulas to be proved are preprocessed by skolemization and by reduction to clausal (resolution-based ATPs) or *if-then-else* (BDD-based ATPs) normal form.
- IPAs should automatize the decidable (and boring) details of proofs such as the use of propositional rules, handling of equality/orderings, expansion of definitions, limited automatic use of simple lemmas (universal Horn clauses), and limited guessing of witnesses.

IPAs work in various logics. CL and ACL^2 [KMM02] work in first-order logic, essentially in primitive recursive arithmetic (PRA). Most of IPAs work with functionals, and many use constructive logic such as type theories. We are concerened here with classical calculi whether one-sorted or many-sorted (possibly with sorts for the functionals of finite types).

A typical work in an IPA is shown in Fig. 1. A theory is extended by a definition of the symbol f and a lemma (n) about it is proved. The lemma is then used in a proof of the theorem (m). The proof of (m) is itself done in the style of extensions. The theory is locally extended (definition hiding) with the symbol g and a local lemma (i) about is is proved. Both lemmas are used to finish the proof of (m).

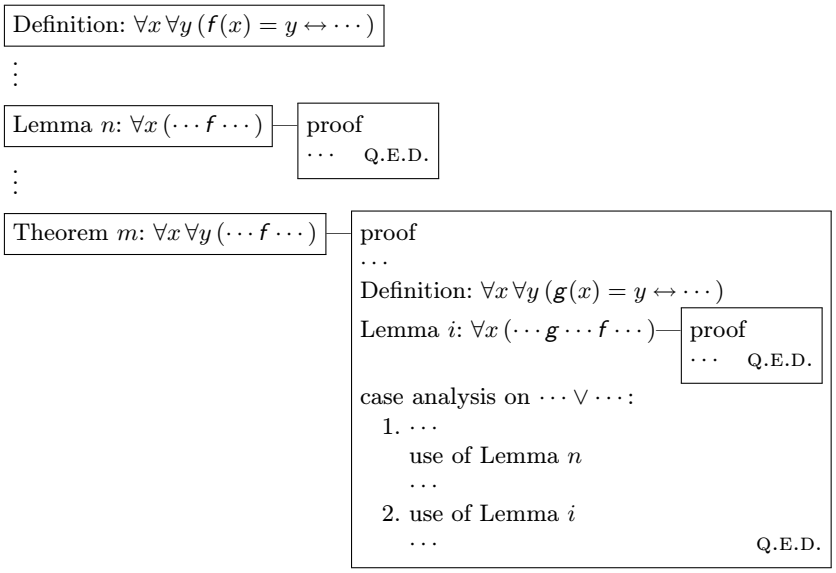


Fig. 1. A schema of development of proofs by a proof assistant in extensions of theories.

3 Formulas

3.1 Formulas and sentences. We use the standard notion of countable languages for first-order logic (denoted by $\mathcal{L}$) with predicate symbols (R, ...) and function symbols (f, g, ...). We denote symbols of any kind by F, G, and sets of symbols by $\mathcal{F}$. We denote by $\mathcal{L}[\mathcal{F}]$ the language $\mathcal{L}$ *extended* with the function and predicate symbols $\mathcal{F}$. We will abbreviate $e_1, \ldots, e_n$ to $\vec{e}$ if n is known from the context of irrelevant.

Terms (t, s, ...) are built up from variables (x, y, z, ...) and applications of function symbols (including constants). *Atomic formulas* are *applications* $R(\vec{s})$ or *equalities* $s = t$. *Formulas* are the least class containing atomic formulas, $\top$, $\bot$, and closed under *existential quantification* $\exists x\,A$ and *if-then-else* construct $\mathrm{If}(A, B, C)$ with A, B, C formulas. *Prime* formulas (designated by P) are atomic or existentially quantified formulas.

We will usually display $\mathrm{If}(A, B, C)$, whose intended interpretation is $(A \wedge B) \vee (\neg A \wedge C)$, as

$$\begin{array}{ccc} & A & \\ \swarrow & & \searrow \\ B & & C \end{array}.$$

We call A the *guard*.

We use the standard notions of free and bound variables. We use $A[x/t]$ to designate the substitution of a closed term t for all free occurrences of the variable x in A. A *sentence* is a closed formula, i.e., a formula with no free variables.

A set of sentences is a *theory* if the codes of its sentences form a recursive set. Theories are denoted by the letters T and S. We write T, S and A, T as abbreviations of $T \cup S$ and $\{A\} \cup T$ respectively.

3.2 Semantics of sentences. We employ the standard notion of a structure $\mathcal{M}$ for a first-order language $\mathcal{L}$. For a fixed structure $\mathcal{M}$ for $\mathcal{L}$, a structure $\mathcal{M}'$ is its *expansion* to a language $\mathcal{L}[\mathcal{F}]$, if the domains of the two structures are identical and $\mathcal{M}'$ coincides with $\mathcal{M}$ on all symbols of $\mathcal{L}$. We denote by $\mathcal{M}[\mathsf{F} \mapsto F]$ the expansion to $\mathcal{L}[\mathsf{F}]$ of the structure $\mathcal{M}$ for $\mathcal{L}$ assigning the function/relation F to the symbol F.

Satisfaction of a sentence A in a structure $\mathcal{M}$, in writing $\mathcal{M} \vDash A$, is defined to satisfy:

(i) the usual properties when A is atomic,
(ii) $\mathcal{M} \vDash \begin{smallmatrix} & A & \\ B & & C \end{smallmatrix}$ iff $\mathcal{M} \vDash A$ and $\mathcal{M} \vDash B$, or if $\mathcal{M} \nvDash A$ and $\mathcal{M} \vDash C$.
(iii) $\mathcal{M} \vDash \exists x\, A$ iff we have $\mathcal{M}[\mathsf{c} \mapsto m] \vDash A[x/\mathsf{c}]$ for some m in the domain of $\mathcal{M}$ and a constant symbol $\mathsf{c} \notin \mathcal{L}$.

A structure $\mathcal{M}$ is a *model* of a theory T, in writing $\mathcal{M} \vDash T$, if $\mathcal{M} \vDash A$ for all $A \in T$. A theory T *logically implies* A, in writing $T \vDash_{\mathcal{L}} A$, if both T and A are in $\mathcal{L}$ and each model of T satisfies A. A sentence A is *valid*, in writing $\vDash A$, if $\emptyset \vDash A$. A theory T is *unsatisfiable* if it has no model, i.e., if $T \vDash \bot$.

3.3 Abbreviations. We will now define the usual propositional connectives and universal quantification without extending the class of formulas. The following definition of the negation connective is possible because the formula $\mathrm{If}(A, B, C)$ has the same semantics as $(A \to B) \wedge (\neg A \to C)$ which is equivalent to $\neg((A \wedge \neg B) \vee (\neg A \wedge \neg C))$:

$$\neg\top :\equiv \bot \qquad \neg\bot :\equiv \top \qquad \neg P :\equiv \begin{smallmatrix} & P & \\ \top & & \bot \end{smallmatrix}$$

$$\neg\Big(\begin{smallmatrix} & A & \\ B & & C \end{smallmatrix}\Big) :\equiv \begin{smallmatrix} & A & \\ \neg B & & \neg C \end{smallmatrix}.$$

The reader will note that *if-then-else* formulas A are in the *negation normal form* for which the law of double negation is a mere syntactic identity: $\neg\neg A \equiv A$. The Tarskian semantics of negation holds for all sentences A because induction on A proves that $\mathcal{M} \vDash \neg A$ iff $\mathcal{M} \nvDash A$.

The usual propositional connectives are defined as follows:

$$A \wedge B :\equiv \begin{smallmatrix} & A & \\ B & & \bot \end{smallmatrix} \qquad A \vee B :\equiv \begin{smallmatrix} & A & \\ \top & & B \end{smallmatrix} \qquad A \to B :\equiv \begin{smallmatrix} & A & \\ B & & \top \end{smallmatrix} \qquad A \leftrightarrow B :\equiv \begin{smallmatrix} & A & \\ B & & \neg B \end{smallmatrix}.$$

It is easy to see that the expected Tarskian properties of satisfaction hold for the defined connectives.

We define the *universally quantified* sentences as $\forall x\, A :\equiv \neg\exists x\, \neg A$ and obtain the expected semantics when $\mathcal{M} \vDash \forall x\, A$ iff for a constant symbol $\mathsf{c} \notin \mathcal{L}$ we have $\mathcal{M}[\mathsf{c} \mapsto m] \vDash A[x/\mathsf{c}]$ for all m in the domain of $\mathcal{M}$.

3.4 Finite sequences of sentences. We will use a computer science *list* notation for finite sequences of sentences. Empty sequence is $\langle\rangle$ and the prefixing of a sequence Π with a sentence A will be designated by the pairing function $\langle A \,|\, \Pi\rangle$. Pairing is such that $\langle\rangle \neq \langle A|\,\Pi\rangle$. We use abbreviations $\langle A, B\,|\,\Pi\rangle := \langle A|\langle B \mid \Pi\rangle\rangle$ and $\langle A_1, \ldots, A_n\rangle := \langle A_1, \ldots, A_n \mid \langle\rangle\rangle$. Concatenation $\oplus$ of sequences is defined as $\langle\rangle \oplus \Phi := \Phi$ and $\langle A \mid \Pi\rangle \oplus \Phi := \langle A \mid \Pi \oplus \Phi\rangle$. All declarative programming languages use lists in such a form. The presented notation is PROLOG's but we have replaced its square brackets with the angle ones.

The sequence Π' *extends* Π if $\Pi' = \Pi \oplus \Phi$ for some Φ. We define

$$\mathrm{Set}(\Pi) = \{A \mid \Pi = \Pi_1 \oplus \langle A \mid \Pi_2\rangle \text{ for some } \Pi_1,\, \Pi_2\}.$$

We adopt a convention that an occurrence of a sequence Π in a position where a set is expected stands for $\mathrm{Set}(\Pi)$. Thus, for instance $A \in \Pi$ stands for $A \in \mathrm{Set}(\Pi)$ and $T, \Pi \vDash \Pi'$ for $T, \mathrm{Set}(\Pi) \vDash \mathrm{Set}(\Pi')$.

Note that $\mathrm{Set}(\Pi \oplus \Pi') = \mathrm{Set}(\Pi) \cup \mathrm{Set}(\Pi')$ and if Π' extends Π then $\Pi \subseteq \Pi'$.

3.5 Paths through sentences. We define a **partial** function $(A)_\Pi$ *selecting a subsentence of* A *at* Π to be defined only if it satisfies:

$$(A)_{\langle\rangle} :\equiv A \qquad (P)_{\langle A\rangle} :\equiv \Big(\begin{smallmatrix} & P & \\ \top & & \bot \end{smallmatrix}\Big)_{\langle A\rangle}$$

$$\Big(\begin{smallmatrix} & A & \\ B & & C \end{smallmatrix}\Big)_{\langle A|\Pi\rangle} :\equiv (B)_\Pi \qquad \Big(\begin{smallmatrix} & A & \\ B & & C \end{smallmatrix}\Big)_{\langle \neg A|\Pi\rangle} :\equiv (C)_\Pi\,.$$

Note that we have $(A)_{\Pi_1 \oplus \Pi_2} \equiv ((A)_{\Pi_1})_{\Pi_2}$ if defined.

The set of all *paths in* A is defined as $\mathrm{Ps}(A) := \{\Pi \mid (A)_\Pi \text{ is defined}\}$. We say that Π is a *path through* A if Π selects $\top$ or $\bot$ in A. We say that A *has a hole at* Π if Π selects $\bot$ in A. A sentence A is *syntactically valid* if it has no holes, i.e., if all paths through A select $\top$.

We define a ternary partial *replacement* function $A[\Pi := D]$ to yield sentences like A but with the subsentence selected by Π (if any) replaced by D. The function is defined only when it satisfies:

$$A[\langle\rangle := D] :\equiv D \qquad P[\langle A\rangle := D] :\equiv \Big(\begin{smallmatrix} & P & \\ \top & & \bot \end{smallmatrix}\Big)[\langle A\rangle := D]$$

$$\Big(\begin{smallmatrix} & A & \\ B & & C \end{smallmatrix}\Big)[\langle A \mid \Pi\rangle := D] :\equiv \begin{smallmatrix} & A & \\ B[\Pi := D] & & C \end{smallmatrix}$$

$$\Big(\begin{smallmatrix} & A & \\ B & & C \end{smallmatrix}\Big)[\langle \neg A \mid \Pi\rangle := D] :\equiv \begin{smallmatrix} & A & \\ B & & C[\Pi := D] \end{smallmatrix}\,.$$

3.6 Lemma.

(i) For every sentence $A \in \mathcal{L}$ *and a structure* $\mathcal{M}$ *for* $\mathcal{L}$*, there is a unique path* Π *through* A *such that* $\mathcal{M} \vDash \Pi$*.*

(ii) $\Pi \vDash A \leftrightarrow (A)_\Pi$ *for every* $\Pi \in \mathrm{Ps}(A)$*.*

(iii) If A is syntactically valid, then $\vDash A$.
(iv) If $T, \Pi \vDash \bot$ for every path Π through D, then $T, D \vDash \bot$.

Proof. (i) By induction on A. If A is $\top$ or $\bot$, the only path $\langle\rangle$ is trivially satisfied by $\mathcal{M}$. If A is prime then if $\mathcal{M} \vDash A$, the path $\langle A\rangle$ is the only one satisfied by $\mathcal{M}$. If $\mathcal{M} \nvDash A$, the path $\langle \neg A\rangle$ is the only one satisfied by $\mathcal{M}$. The case when A is an *if-then-else* sentence follows from IH.

(ii) Observe first that if $\Pi = \langle\rangle$, then there is nothing to prove because $(A)_{\langle\rangle} \equiv A$. For $\Pi \neq \langle\rangle$, A cannot be $\top$ or $\bot$, and we continue by induction on A. If A is prime, then the conclusion follows if $\Pi = \langle A\rangle$, since $(A)_\Pi \equiv \top$. The case $\Pi = \langle \neg A\rangle$ is similar because then $(A)_\Pi \equiv \bot$. If $A \equiv \overset{B}{C}\diagdown\diagup_{D}$, then we must have $\Pi = \langle B \,|\, \Pi'\rangle$ with $\Pi' \in \mathrm{Ps}(C)$ or $\Pi = \langle \neg B \,|\, \Pi'\rangle$ with $\Pi' \in \mathrm{Ps}(D)$. If the former, then for any $\mathcal{M} \vDash \Pi$ we have $\mathcal{M} \vDash C \leftrightarrow (C)_{\Pi'}$ from IH. Since $\mathcal{M} \vDash B$, we have $\mathcal{M} \vDash A \leftrightarrow C$ and the conclusion follows because $(A)_\Pi \equiv (C)_{\Pi'}$. The other case is similar.

(iii) Take any $\mathcal{M}$. By (i) there is a path Π through A such that $\mathcal{M} \vDash \Pi$. Since $(A)_\Pi \equiv \top$, we get $\mathcal{M} \vDash A$ by (ii).

(iv) Suppose $\mathcal{M} \vDash T, D$ for some $\mathcal{M}$. By (i), $\mathcal{M} \vDash \Pi$ for some path Π through D. This falsifies the assumption. □

4 Proofs

If a sentence A is valid, then all paths to a hole of A must be inconsistent. A proof of A will be a sentence D "plugging all holes in A". D will be obtained from A by replacing its holes with sentences which decidably cannot be satisfied.

4.1 Regular sets of extension axioms. Sentences of the form $\exists x\, A \to A[x/\mathsf{c}]$ *defining* the constant c which does not occur in A are *Henkin* axioms. Sentences of the form $\forall \vec{x}\,(\mathsf{R}(\vec{x}) \leftrightarrow A)$ *defining* the predicate symbol R which does not occur in A are *defining axioms for* R. Sentences of the form $\forall \vec{x}\, \exists! y\, A \to \forall \vec{x} \forall y\,(\mathsf{f}(\vec{x}) = y \leftrightarrow A)$ *defining* the function symbol f which does not occur in A are *defining axioms for* f. All three kinds of axioms are called *extension axioms*.

Let $\mathcal{L}$ be a language, and $\mathcal{F}$ a countable set of symbols not in $\mathcal{L}$. A set S of extension axioms in $\mathcal{L}[\mathcal{F}]$ is *regular for* $\mathcal{F}$ if S can be enumerated without repetitions as $\{S_i\}_{i=1}^{|S|}$ where each extension axiom S_i defines a symbol $\mathsf{F}_i \in \mathcal{F}$ not occurring in $\{S_1, \ldots, S_{i-1}\}$.

4.2 Syntactic inconsistency. Let T be a theory in $\mathcal{L}$ and S a set of extension axioms in $\mathcal{L}[\mathcal{F}]$ regular for $\mathcal{F}$. We designate by $X(T, S)$ the theory in $\mathcal{L}[\mathcal{F}]$ consisting of the following sentences:

(1) $\overset{A[x/t]}{\underset{\top \quad \bot}{\exists x\, \overset{A}{\diagup\diagdown}}}\ \top$, (2) $\overset{t=t}{\top \quad \bot}$, (3) $\overset{s=t}{\underset{P[x/t]}{\underset{\top \quad \bot}{P[x/s]}}}\ \top$ for atomic P,

(4) $\overset{\exists x\, A}{\diagup\ \diagdown}$ $A[x/c]$ $\top$ for all $(\exists x\, A \to A[x/c]) \in S$, (5) $\overset{A}{\top\ \ \bot}$ for all $A \in T$,

(6) $\overset{\forall \vec{x}\,(R(\vec{x}) \leftrightarrow A)}{\top\ \ \bot}$ for all $(\forall \vec{x}\,(R(\vec{x}) \leftrightarrow A)) \in S$,

(7) $\forall \vec{x} \forall y\,(f(\vec{x}) = y \overset{\forall \vec{x}\, \exists! y\, A}{\leftrightarrow} A)\ \top$ for all $(\forall \vec{x}\, \exists! y\, A \to \forall \vec{x} \forall y\,(f(\vec{x}) = y \leftrightarrow A)) \in S$.
$\top\ \ \bot$

We say that a path Π in $\mathcal{L}[\mathcal{F}]$ is *syntactically inconsistent with* T *and* S if there is a hole at $\Pi' \subseteq \Pi$ in some $A \in \Pi, X(T, S)$.

Let $\mathcal{M}$ be a model of T and S. The sentences (4)–(7) are equivalent to the corresponding axioms from T and S, and are thus satisfied by $\mathcal{M}$. The sentences (1)–(3) are equivalent in that order to $A[x/t] \to \exists x\, A$, $t = t$, and $s = t \wedge P[x/s] \to P[x/t]$, and they are satified in any structure. Thus $\mathcal{M} \vDash X(T, S)$.

For a sentence A in $\mathcal{L}$ with a hole at Π we say that a syntactically valid sentence D in $\mathcal{L}[\mathcal{F}]$ *plugs the hole in* A *at* Π *w.r.t.* T *and* S if for every path Π' through D, the path $\Pi \oplus \Pi'$ is syntactically inconsistent with T and S.

4.3 Proofs. A sentence D in a language $\mathcal{L}[\vec{F}]$ is a *proof* witnessing $T \vDash_{\mathcal{L}} A$, in writing $D \vdash_{\vec{F}} T \vDash_{\mathcal{L}} A$, if

(i) there is a regular set S in $\mathcal{L}[\mathcal{F}]$ of extension axioms defining all symbols $\vec{F}$,
(ii) all holes of A are at $\Pi_1, \ldots, \Pi_n$ and to every $i = 1, \ldots, n$ there is a D_i in $\mathcal{L}[\vec{F}]$ plugging the hole at Π_i w.r.t. T and S,
(iii) we have $D \equiv A[\Pi_1 := D_1] \cdots [\Pi_n := D_n]$.

Note that D is syntactically valid.

We must show that the property $D \vdash_{\vec{F}} T \vDash_{\mathcal{L}} A$ is decidable. The paths Π_i and the plugs D_i can be recovered from A and D. We must be able to test every path Π through D extending some Π_i for syntactic inconsistency. There are only finitely many paths $\Pi' \subseteq \Pi$ from the definition of syntactic inconsistency in Par. 4.2. Every such Π' can be tested for leading to a hole of finitely many sentences $A \in \Pi$. Failing this, Π' must lead to a hole in one of the sentences (1)–(7). To test whether Π' leads to a hole in (1)–(3) is easy. To see whether Π' leads to a hole in some axiom T, we observe that then $\Pi' \equiv \langle A \rangle$ for some A and $\neg A$ must be in T. This is decidable, since T is recursively coded. The extension axioms can be fully determined from Π'. There are only finitely many candidates for the (finite) sets of extension axioms causing inconsistency in D and they can be tested for regularity for $\vec{F}$.

We write $T \vdash_{\mathcal{L}} A$ if T and A are in $\mathcal{L}$, and $D \vdash_{\vec{F}} T \vDash_{\mathcal{L}} A$ for some D and $\vec{F}$.

4.4 Lemma. *If* T *is a theory in* $\mathcal{L}$ *and* S *a set of extension axioms regular for* $\mathcal{F}$ *such that* $\vec{F} \subseteq \mathcal{F}$*, then*

(i) any model of T *can be expanded to a model of* S*,*

(ii) if Π in $\mathcal{L}[\vec{F}]$ is syntactically inconsistent with T and S then $T, S, \Pi \vDash \bot$.

Proof. (i) Let $\{S_i\}_{i=1}^{|S|}$ be as in Par. 4.1 and let $\mathcal{M}$ for $\mathcal{L}$ be a model of T. We form a sequence $\{\mathcal{M}_i\}_{i=0}^{|S|}$ of structures $\mathcal{M}_i$ for $\mathcal{L}[\mathsf{F}_1, \ldots, \mathsf{F}_i]$ where $\mathcal{M}_0 = \mathcal{M}$, and for $i > 0$, $\mathcal{M}_i$ is an expansion of $\mathcal{M}_{i-1}$ which satisfies the extension axiom S_i defining F_i. Note that such $\mathcal{M}_i$ can be always found, even if S_i is a Henkin axiom or an axiom introducing a function symbol, and we respectively have $\mathcal{M}_{i-1} \nvDash \exists x\, A$ or $\mathcal{M}_{i-1} \nvDash \forall \vec{x}\, \exists! y\, A$. We then simply choose arbitrary interpretation of F_i. The desired model of S is the union of structures $\mathcal{M}_i$.

(ii) Suppose the contrary, i.e., that $\mathcal{M} \vDash T, S, \Pi$. Then also $\mathcal{M} \vDash X(T, S)$. From the assumption, there is $A \in \Pi, X(T, S)$ with a hole at some $\Pi' \subseteq \Pi$. By Lemma 3.6(ii), we have $\mathcal{M} \vDash A \leftrightarrow (A)_{\Pi'}$. But this is a contradiction because $\mathcal{M} \vDash A$ and $(A)_{\Pi'} \equiv \bot$. □

4.5 Theorem (Soundness). *If $T \vdash_{\mathcal{L}} A$, then $T \vDash A$.*

Proof. Let $D \vdash_{\vec{F}} T \vDash_{\mathcal{L}} A$ with S, $\Pi_1, \ldots, \Pi_n$, and $D_1, \ldots, D_n$ as in Par. 4.3. All sentences D_i are valid by Lemma 3.6(iii). Take any model $\mathcal{M}$ of T and expand it using Lemma 4.4(i) to a model $\mathcal{M}'$ of S. By Lemma 3.6(i), there is a path Π_0 through A s.t. $\mathcal{M}' \vDash \Pi_0$. The theorem will be proved if we contrive to show that $(A)_{\Pi_0} \equiv \top$, because then by (ii) of the same lemma, we will have $\mathcal{M}' \vDash A$, and hence $\mathcal{M} \vDash A$.

So assume, in a way of contradiction, that $(A)_{\Pi_0} \equiv \bot$. We must then have $\Pi_0 \equiv \Pi_i$ for some $i = 1, \ldots, n$. Take any path Π through D_i. Since D_i plugs a hole at Π_0, the path $\Pi_0 \oplus \Pi$ is syntactically inconsistent with T and S. Thus $\Pi_0, T, S, \Pi \vDash \bot$ by Lemma 4.4(ii). Lemma 3.6(iv) now applies, and we get $\Pi_0, T, S, D_i \vDash \bot$. We now have a contradiction $\mathcal{M}' \nvDash \Pi_0$ because of $\mathcal{M}' \vDash T, S, D_i$. □

5 Completeness

We will prove completeness of our proof calculus by the simplest possible method. We will construct a sequence of sentences $\{D_i\}_{i \in \mathbb{N}}$ such that $D_0 :\equiv A$ where A is the sentence to be proved. The sentence D_{i+1} will be obtained from D_i by plugging every hole in D_i at a syntactically inconsistent path with $\top$ and every other hole by a sentence $\overset{B_i}{\bot}\!\!\searrow\!\!\underset{\bot}{}$. With a suitable enumeration of sentences $\{B_i\}_{i \in \mathbb{N}}$, we either stumble by this, rather brute-force, method at a syntactically valid D_i which will be a proof of A, or else we will be able to find an infinite path H through all sentences D_i starting at some hole at Π in A such that some $\mathcal{M}$ is a model of H. As also $\mathcal{M} \vDash \Pi$, it will follow that $\mathcal{M} \nvDash A$.

5.1 Witnessing expansion of $\mathcal{L}$. The *witnessing expansion* of the language $\mathcal{L}$ is a language $\mathcal{L}[\mathcal{C}]$, in which there is a *Henkin witnessing* constant $\mathsf{c}_{\exists x A} \in \mathcal{C}$ for each sentence $\exists x\, A$ in $\mathcal{L}[\mathcal{C}]$. The assignment of constants is such that the set

$$He := \{\exists x\, A \to A[x/\mathsf{c}_{\exists x A}] \mid \exists x\, A \in \mathcal{L}[\mathcal{C}]\}$$

of Henkin axioms is regular for $\mathcal{C}$.

Construction of the witnessing expansion is given, e.g., in [Bar77].

5.2 Hintikka sets. A theory H in $\mathcal{L}$ *simply consistent* if for each sentence $A \in H$ we have $(\neg A) \notin H$. A simply consistent theory H is *maximal* if it has no proper simply consistent extension in $\mathcal{L}$, i.e., if for all $A \in \mathcal{L}$ either A or $\neg A$ is in H.

Let $\mathcal{L}[\mathcal{C}]$ be the witnessing expansion of $\mathcal{L}$. A theory H in $\mathcal{L}[\mathcal{C}]$ is called a *Hintikka set* if the following conditions are satisfied:

(i) $\top \in H$, and $(t = t) \in H$ for all terms t in $\mathcal{L}[\mathcal{C}]$.
(ii) If $(s = t) \in H$ and $P[x/s] \in H$ for atomic P, then $P[x/t] \in H$.
(iii) If ${}_{B}{\nearrow}^{A}{\searrow}_{C} \in H$, then $A \in H$ and $B \in H$, or $(\neg A) \in H$ and $C \in H$.
(iv) If $\exists x\, A \in H$, then $A[x/\mathsf{c}_{\exists x A}] \in H$.
(v) If $\forall x\, A \in H$, then $A[x/t] \in H$ for all terms t in $\mathcal{L}[\mathcal{C}]$.

5.3 Lemma. *Every maximal simply consistent Hintikka set has a model.*

Proof. For a similar construction see, e.g., [Bar77,Sho67]. □

5.4 Brute-force proof construction. Let A be a sentence, T a theory, both in $\mathcal{L}$, and He be as in Par. 5.1. To any pair of sentences D, B in $\mathcal{L}[\mathcal{C}]$ we define a sentence $\mathrm{BF}(D, B)$ which extends D with the sentence ${}_{\bot}{\nearrow}^{B}{\searrow}_{\bot}$ at its syntactically consistent holes:

$$\mathrm{BF}(D, B) :\equiv D[\Pi_1 := E_1] \cdots [\Pi_n := E_n]$$

where $\Pi_1, \ldots, \Pi_n$ $(n \geq 0)$ are paths to all holes in D, and for $j = 1, \ldots, n$

$$E_j :\equiv \begin{cases} \top & \text{if } \Pi_i \text{ is syntactically inconsistent with } T \text{ and } He, \\ {}_{\bot}{\nearrow}^{B}{\searrow}_{\bot} & \text{otherwise.} \end{cases}$$

For an enumeration $\{B_i\}_{i=0}^{\infty}$ of all sentences in $\mathcal{L}[\mathcal{C}]$ we define a sequence of sentences $\{D_i\}_{i=0}^{\infty}$ by $D_0 :\equiv A$ and $D_{i+1} :\equiv \mathrm{BF}(D_i, B_i)$.

5.5 Theorem (Completeness). *If $T \vDash_{\mathcal{L}} A$, then $T \vdash_{\mathcal{L}} A$.*

Proof. Assume $T \vDash_{\mathcal{L}} A$, construct the sequence $\{D_i\}_{i=0}^{\infty}$ as in Par. 5.4, and let $\Pi_1, \ldots, \Pi_n$ be paths leading to all holes in A. A straightforward induction on i proves:

> *For each $i > 0$ and each path Π through D_i extending some Π_j, we have $(D_i)_{\Pi} \equiv \top$ iff Π is syntactically inconsistent with T and He.*

We consider two cases. If $D_i \equiv D_{i+1}$ for some i, then D_i has no holes, and so the sentences $(D_i)_{\Pi_1}, \ldots, (D_i)_{\Pi_n}$ are syntactically valid. Since by the above property, each $(D_i)_{\Pi_j}$ plugs the hole in A at Π_j w.r.t. T and He, we have $D_i \vdash_{\vec{F}} T \vDash_{\mathcal{L}} A$.

If all D_i are different, we define the sequence $\{\mathcal{T}_i\}_{i=0}^{\infty}$ by

$$\mathcal{T}_0 := \{\Pi \mid A \text{ has a hole at } \Pi' \text{ and } \Pi' \text{ extends } \Pi\}$$
$$\mathcal{T}_{i+1} := \{\Pi \mid D_i \text{ has a hole at } \Pi\} \cup \mathcal{T}_i,$$

and let $\mathcal{T} := \bigcup_{i=0}^{\infty} \mathcal{T}_i$. The set $\mathcal{T}$ is infinite and such that if $\Pi' \in \mathcal{T}$ and Π' extends Π, then $\Pi \in \mathcal{T}$. This makes $\mathcal{T}$ a (set-theoretical) tree. It is finitely branching because every $\Pi \in \mathcal{T}$ has at most two immediate successors (extensions) in $\mathcal{T}$. By König's lemma, $\mathcal{T}$ has an infinite branch $\Phi_0 \subseteq \Phi_1 \subseteq \Phi_2 \subseteq \cdots \subseteq \mathcal{T}$. From the construction of $\mathcal{T}$, all Φ_i are syntactically consistent (with T and He). We set $H := \bigcup_{i=0}^{\infty} \Phi_i$ with the intention of showing that H is a maximal simply consistent Hintikka set s.t. $T \subseteq H$. By Lemma 5.3, there will be then a structure $\mathcal{M} \vDash H$. Since $\Phi_j = \Pi_k$ for some j and k, we will get $\mathcal{M} \nvDash A$ from Lemma 3.6(ii). The retraction $\mathcal{M}'$ to $\mathcal{L}$ of $\mathcal{M}$ will contradict the assumption $T \vDash_{\mathcal{L}} A$.

We first show that H *is maximal* in $\mathcal{L}[\mathcal{C}]$ by taking any $B \in \mathcal{L}[\mathcal{C}]$. It is some B_i in the enumeration of $\mathcal{L}[\mathcal{C}]$ and thus $\Phi_{j+1} = \Phi_j \oplus \langle B \rangle$ or $\Phi_{j+1} = \Phi_j \oplus \langle \neg B \rangle$ for some $j \geq i$. Hence one of B, $\neg B$ is in H.

We will need two auxiliary properties of H:

No $\Pi \subseteq H$ is a path to a hole in a sent. (1)–(5) of Par. 4.2 or in an $A' \in H$. (†)

For every sentence $B \in \mathcal{L}[\mathcal{C}]$ there is a path Π through B s.t. $\Pi \subseteq H$. (∗)

(†) Assume $\Pi \subseteq H$. If Π is a path to a hole in a sentence (1)–(5), then $\Pi \subseteq \Phi_j$ for some j, and Φ_j is syntactically inconsistent. If Π is a path to a hole in A', then $A', \Pi \subseteq \Phi_j$ for some j, and Φ_j is syntactically inconsistent.

(∗) We proceed by induction on B. If B is $\top$ or $\bot$, then set $\Pi := \langle\rangle \subseteq H$. If B is prime, then by maximality B or $\neg B$ is in H, and so set $\Pi := \langle B \rangle \subseteq H$ in the first case, and $\Pi := \langle \neg B \rangle \subseteq H$ in the second. If $B \equiv {}_{D}{}^{C}\!\diagdown_{E}$, then C or $\neg C$ is in H by maximality. In the first case, set $\Pi := \langle C \mid \Pi' \rangle$ where $\Pi' \subseteq H$ is obtained from IH as a path through D. The second case is similar.

We will now show that H *is simply consistent*. Assume that both B and $\neg B$ are in H. By (∗), there is a path Π through B s.t. $\Pi \subseteq H$. Since the same Π is also a path through $\neg B$, Π leads to a hole of exactly one of B or $\neg B$. This contradicts (†) where A' is B or $\neg B$.

We will now show that T *is a subset of* H. Take any $B \in T$. $\langle \neg B \rangle \subseteq H$ violates (†)(5). Hence $B \in H$ by the maximality of H.

It remains to show that H *is a Hintikka set*:

(i) Since $\langle\rangle \subseteq H$, and it is a path to a hole of $\bot$, $\bot \in H$ would violate (†) with $A' \equiv \bot$. Hence $\top \in H$ by the maximality of H. We cannot have $\langle t \neq t \rangle \subseteq H$ by (†)(2). Hence $(t = t) \in H$ by the maximality of H.

(ii) If $(s = t), P[x/s] \in H$ then, since $\langle s = t, P[x/s], \neg P[x/t] \rangle \subseteq H$ would violate (†)(3), we must have $P[x/t] \in H$ by the maximality of H.

(iii) If $A' :\equiv {}_{C}{}^{B}\!\diagdown_{D} \in H$, then there is a path $\Pi \subseteq H$ through A' by (∗). By (†), Π cannot lead to a hole of A'. Exactly one of B, $\neg B$ is in H. If the former, then $\Pi = \langle B \mid \Pi' \rangle$ with Π' a path through C. Π' cannot be a path to a hole of C because then A' would have a hole at Π. Hence Π' is a path to a hole

of $\neg C$. By (†), we must have $(\neg C) \notin H$. Hence $C \in H$ by the maximality of H. The case when $(\neg B) \in H$, which implies $D \in H$, is similar.

(iv) If $\exists x\, B \in H$ then, since $\langle \exists x\, B, \neg B[x/\mathsf{c}_{\exists x B}] \rangle \subseteq H$ would violate (†)(4), we must have $B[x/\mathsf{c}_{\exists x B}] \in H$ by the maximality of H.

(v) For any t, if $\forall x\, B \equiv \neg\exists x \neg B \in H$, then, since $\langle \neg B[x/t], \neg\exists x \neg B \rangle \subseteq H$ would violate (†)(1), we must have $B[x/t] \in H$ by the maximality of H. □

6 Methods of Proof Construction

We have introduced a new proof calculus and demonstrated that its concept of provability coincides, as it should, with logical consequence. It remains to convince the reader that the calculus offers advantages over the standard proof systems. In this section we will show that

- we can construct proofs incrementally by applying local proof rules,
- the rules are flexible enough to emulate Hilbert, as well as Gentzen style of proofs,
- we can, because of definition hiding, transform proofs with flexibility not afforded by the last two types of calculi.

6.1 Derivation rules. For the duration of this section we fix a (possibly empty) theory T in some $\mathcal{L}$. A *derivation rule* (R) is of the form

$$(R)\ \frac{\Gamma}{D}$$

where Γ is a finite set of sentences (possibly empty) in $\mathcal{L}$, called *premises*, and the sentence D in $\mathcal{L}[\vec{F}]$ is the *conclusion*. The rule is *sound* if the following is constructively proved:

> for all paths Π such that $\Gamma \subseteq \Pi$, and every A in $\mathcal{L}[\vec{F}]$ with a hole at Π and such that $T \vdash_{\mathcal{L}[\vec{F}]} A[\Pi := D]$, we have $T \vdash_{\mathcal{L}} A$.

For the soundness of the rule (R) it suffices that D is such that every path Π' selecting $\top$ in D (if any) leads to inconsistency: $T, S, \Gamma, \Pi' \vdash_{\mathcal{L}[\vec{F}]} \bot$ for some S. This is an easy consequence of the definition of proofs. The simplest proof of the last condition is $\top$ when $\Pi \oplus \Pi'$ is syntactically inconsistent with T and S.

6.2 The case analysis (cut) method. The *case analysis* (cut) method is characterized by two rules:

$$\text{(Close)}\quad \frac{\Gamma}{\top} \quad \text{if } \Gamma \text{ is synt. inconsistent with } T \text{ and some } S$$

$$\text{(Cut)}\quad \frac{}{\begin{array}{c} A \\ \diagup\ \diagdown \\ \bot \quad \bot \end{array}}$$

Soundness of the two rules follows from the sufficient condition in Par. 6.1.

6.3 The unboxing method. The *unboxing* method takes sentences out of the guards of *if-then-else* connectives, possibly also by dropping quantifiers, and plugs holes with them. Its rules are (Close) (see Par. 6.2), the rule "deeply" *unboxing* a guard (Unbox), and five "shallowly" unboxing rules: (Th) for axioms in T, rules ($L\exists$), ($L\forall$) for quantifiers, and the two rules for equality (Id), (Subst).

$$\text{(Unbox)}\ \frac{A}{\neg A} \qquad \text{(Th)}\ \frac{}{\begin{array}{c}A\\ \bot\quad\top\end{array}}\ \text{if } A\in T.$$

$$\text{(L}\forall\text{)}\ \frac{\forall x\, A}{\begin{array}{c}A[x/t]\\ \bot\quad\top\end{array}} \qquad \text{(L}\exists\text{)}\ \frac{\exists x\, A}{\begin{array}{c}A[x/\mathsf{c}]\\ \bot\quad\top\end{array}}\ \text{if } \mathsf{c}\notin\mathcal{L}.$$

$$\text{(Id)}\ \frac{}{\begin{array}{c}t=t\\ \bot\quad\top\end{array}} \qquad \text{(Subst)}\ \frac{s=t, P[x/s]}{\begin{array}{c}P[x/t]\\ \bot\quad\top\end{array}}\ \text{if } P \text{ is atomic.}$$

The rules are sound because they are reducible to those of the cut method. We can, namely, construct all sentences in conclusions by one or more cuts. What must be justified, is the correctness of applications of (Close) rules in the conclusions of the rules.

So, let Π be the path to the hole we are trying to plug with one of the listed rules. For (Unbox), let Π' be a path selecting any $\top$ in its conclusion $\neg A$. The path $\Pi \oplus \Pi'$ is syntactically inconsistent on account of $A \in \Pi$ and Π' being a path to a hole of A. We show only the soundness of (L$\forall$), the remaining rules are dealt with similarly. The path to the $\top$ in the conclusion of (L$\forall$) is $\Pi \oplus \langle \neg A[x/t] \rangle$. It is inconsistent by containing a path to the hole of (1) of Par. 4.2 because $\forall x\, A \equiv \neg\exists x\, \neg A \in \Pi$.

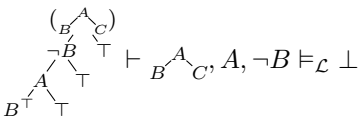
Henkin axioms in the set S of extension axioms can be constructed "on the fly" with every application of (L$\exists$). The same is achieved in the usual calculi by eigen-variable conditions.

6.4 The sequent method. We will now show that our calculus can emulate proofs in Gentzen's sequent calculi. For that we need two rules:

$$\text{(Ax)}\ \frac{P, \neg P}{\top}\ \text{for atomic } P \qquad \text{(If)}\ \frac{\begin{array}{c}A\\ B\quad C\end{array}}{\begin{array}{c}A\\ B\quad\quad C\\ \bot\quad\top\quad\bot\quad\top\end{array}}.$$

The closure in (Ax) is justified by $\langle \neg P \rangle$ being the path to the hole of P. With the same premise as in (If), we would get by (Unbox) the conclusion $\begin{array}{c}A\\ \neg B\quad \neg C\end{array}$ where the sentences $\neg B$ and $\neg C$ are unboxed. The rule (If) gives equivalent, but boxed sentences. The two occurrences of $\top$ in its conclusion are not directly reducible

to (Close). However, one can easily see that



where $B^\top$ is just like B with all holes replaced by $\top$. There is an similar witness of $\begin{smallmatrix} & A & \\ B & & C \end{smallmatrix}, \neg A, \neg C \vdash_{\mathcal{L}} \bot$. This is by Par. 6.1 sufficient for soundness of the rule.

The sequent rules are (Ax), (If), (Th), (L$\forall$), (L$\exists$), (Id), (Subst).

We now present a translation of a proof D of $T \vdash A$ with all inferences in D by sequent rules to a one-sided sequent calculus with sequents $\Gamma \Rightarrow$. Such sequents are dual to the well-known Schütte-Tait one-sided sequents $\Rightarrow \Delta$ in the calculus called **GS** in [TS00]. Assume that a sequent rule $\dfrac{\Gamma}{C}$ has been applied at a path $\Pi \in \mathrm{Ps}(D)$ leading through a hole of A. We translate it to the sequent calculus inference

$$\frac{\Pi_1, \Gamma, \Pi \Rightarrow \quad \ldots \quad \Pi_n, \Gamma, \Pi \Rightarrow}{\Gamma, \Pi \Rightarrow}$$

where $\Pi_1, \ldots, \Pi_n$ are all paths to holes in C. For instance, sequent calculus inferences corresponding to (Ax), (If), and (L$\forall$) are respectively:

$$\text{(Ax)}\ \frac{}{P, \neg P, \Pi \Rightarrow} \quad \text{(If)}\ \frac{A, B, \Pi \Rightarrow \qquad \neg A, C, \Pi \Rightarrow}{\begin{smallmatrix} & A & \\ B & & C \end{smallmatrix}, \Pi \Rightarrow} \quad \text{(L}\forall\text{)}\ \frac{A[x/t], \Pi \Rightarrow}{\forall x\, A, \Pi \Rightarrow}.$$

6.5 Considerations of completeness. In this section we have concentrated on the soundness of derivation rules. We know that the cut method is also complete, because we have proved the completeness theorem 5.5 by its two rules. Incidentally, we have proved more than completeness, we have also semantically demonstrated the conservativity of extensions by definitions because we never had to use other extension axioms than Henkin's. It can be shown that also the unboxing and sequent methods are complete in this sense.

When it comes to proving sentencess A in $\mathcal{L}[\vec{F}]$ by definition hiding, where the extension axioms defining the symbols $\vec{F}$ are local to the proofs, then we cannot hope for the completeness. This is because A can express a property of $\vec{F}$ not satisfiable by any extensions by definitions.

6.6 Inversion. The rule (L$\forall$) in the one-sided sequent calculus is not *invertible*. This means that given a proof $\mathcal{D}$ of the conclusion we cannot in general eliminate all its uses with terms $t_1, \ldots, t_n$ in $\mathcal{D}$ and obtain a proof of $A[x/t_1], \ldots, A[x/t_n], \Gamma \Rightarrow$. This is because the terms t_i may contain eigenvariables introduced by (L$\exists$) in $\mathcal{D}$.

In our calculus we have weakened the eigen-variable conditions to the requirement of regularity of extension axioms. This allows arbitrary permutations where we can, for instance, replace $\begin{smallmatrix} & A & \\ D_1 & & B \\ & D_2 & & D_3 \end{smallmatrix}$ anywhere in $\mathcal{D}$ by $\begin{smallmatrix} & & B & & \\ & A & & & A \\ D_1 & & D_2 & D_1 & & D_3 \end{smallmatrix}$. Thus, given a proof of $\exists x\, A$, we can permute all applications of (L$\forall$) to $\forall x\, \neg A$ in the proof to obtain a proof of $\bigvee_i A[x/t_i]$. Note that this is an example of

definition hiding where we assert a property of Henkin constants occurring in the terms t_i without exposing their defining axioms.

6.7 Future work. We are preparing a paper in which we will demonstrate the usefulness to pure logic of our calculus, with its emphasis on paths through formulas and definition hiding. The paper will be on the still difficult, and not yet completely satisfactorily solved, Π^0_2-analysis of some theories T. The analysis tries to characterize the witnessing functions $\vec{f}$ from a given proof of a sentence $\forall\vec{x}\,\exists\vec{y}\,R(\vec{x},\vec{y})$ with R quantifier-free. The functions satisfy $R(\vec{x},\vec{f}(\vec{y}))$. This is equivalent to determining the ordinal bounds on such proofs.

The prepared paper will deal with predicate calculus ($T=\emptyset$) and with Peano Arithmetic (PA). In both cases, the Henkin witnessing constants introduced from the descendants of cut formulas will be replaced by explicit quantifier-free definitions of witnessing functions. In case of PA, also the induction axioms with quantified formulas will be replaced by quantifier-free definitions of μ-functions yielding the least witnesses.

References

[Ane90] Anellis, I.H.: *From Semantic Tableaux to Smullyan Trees: A History of the Development of the Falsifiability Tree Method.* Modern Logic 1(1):36–69, 1990.

[Bar77] Barwise, J.: First-order logic. In Barwise, J., ed.: *Handbook of Mathematical Logic.* North Holland, 1977.

[BC04] Bertot, Y., Castéran, P.: *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions.* Texts in Theoretical Computer Science. An EATCS series. Springer, 2004.

[Bry86] Bryant, R.E.: *Graph-Based Algorithms for Boolean Function Manipulation.* IEEE Transactions on Computers, C-35(8):677–691, 1986.

[CL97] Clausal Language, `http://ii.fmph.uniba.sk/cl/`

[Gou94] Goubault, J.: Proving with BDDs and Control of Information. In *Proc. 12th Intl. Conf. on Automated Deduction (CADE'94)*, pp. 499–513, Springer, 1994.

[GT03] Groote, J.F., Tveretina, O.: *Binary decision diagrams for first-order predicate logic.* J. Logic and Algebraic Programming 57(1–2):1–22, 2003.

[KMM02] Kaufmann, M., Manolios, P., Moore, J.S.: *Computer-Aided Reasoning: An Approach.* Kluwer Academic Publishers, 2000.

[KV09] Kľuka, J., Voda, P.J.: *A Simple and Practical Valuation Tree Calculus.* Computation in Europe (CiE) 2009. `http://dai.fmph.uniba.sk/~voda/ext-proof.pdf`

[NPW02] Nipkow, T., Paulson, L.C., Wenzel, M.: *Isabelle/HOL — A Proof Assistant for Higher-Order Logic.* LNCS 2283, Springer, 2002.

[PS95] Posegga, J., Schmitt, P.H.: *Automated Deduction with Shannon Graphs.* J. Logic and Computation, 5(6):697–729, 1995.

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

[Sho67] Shoenfield, J.R.: *Mathematical Logic.* Addison-Wesley, 1967.

[TS00] Troelstra, A.S., Schwichtenberg, H.: *Basic Proof Theory. Second edition.* Cambridge University Press, 2000.