A DECISION PROCEDURE FOR PURE FIRST-ORDER LOGIC

TIMM LAMPERT

Humboldt University Berlin, Unter den Linden 6, D-10099 Berlin
e-mail address: lampertt@staff.hu-berlin.de

Abstract. This paper explains the logical foundations, proof strategies and termination criteria of the program "FOL-Decider" that decides for every formula of pure first-order logic without identity whether it is refutable. It is proven that the FOL-Decider is correct and terminates.

Contents

1. Introduction

This paper explains a decision procedure for first-order logic formulas without names, functions or identity (FOL) and proves its validity.[1] The FOL-Decider is an implementation of the algorithm described in this paper, with the purpose of demonstrating the algorithm's

[1]For the discussion of foundational questions concerning the possibility of such a decision procedure cf. [Lampert (2019)].

feasibility. It is programmed in the Wolfram Language of *Mathematica*. This language permits a rather plain and transparent implementation. Only a slight effort has been made to optimize the decision procedure. This is because the intention is for the FOL-Decider to serve as a proof of concept (PoC) that focuses on the logical principles of the decision procedure rather than on its optimization. For this reason, the FOL-Decider is, in general, still slower than well-known logic engines such as Beagle, Darwin, i-Prover, E, Vampire and SPASS. Short formulas with infinite models only, however, constitute a verifiable exception (cf. p. 38). The FOL-Decider accepts the TPTP syntax and runs under the TPTP site. It was tested with all FOL formulas in the TPTP library with fewer than 20 atoms, as well as other formulas. The restriction to formulas with fewer than 20 atoms is due to the disadvantageous properties of the FOL-Decider with regard to fast decision-making.

The general approach of the FOL-Decider is nothing special. Like all logic engines, it intends to refine the search for proofs within a correct and complete calculus to the effect that redundant and unnecessary application of deductive rules are avoided whenever possible. At best, satisfiability can be proven by showing that an exhausted (fully saturated) optimized proof search. The problem with any optimization of the proof search is to preserve correctness and completeness. Termination must not be achieved on the cost of these properties; this has to be proven in the case of any optimization of the proof search. Given the proof of the Church-Turing theorem is faulty, one cannot say in advance whether the endeavour to optimize proof search strategies has a principle limit. According to my point of view, saturation algorithms, which already prove satisfiability in many cases due to an optimized proof strategy, are on the way to a full decision procedure. The FOL-Decider can be conceived as a special version of a saturation algorithm, which provides sufficient principles to decide FOL based on a new proof strategy.

In contrast to powerful logic engines, the FOL-Decider does not make use of resolution and unification algorithms based on skolemization. I have attempted to translate the basic ideas of my decision procedure into linear resolution with subsumption. The proof search strategy used in the FOL-Decider is designed to detect superfluous repetitions of isomorphic proof steps in a proof search. The problem is to define precisely what constitutes a similar superfluous repetition of an isomorphic proof step in the case of a proof search within the resolution calculus for FOL. In a proof search based on resolution, (i) the lengths of resolvents and (ii) the depths of nested skolem functions may increase to infinity. There is no equivalent to (i) within the proof strategy of the FOL-Decider. Regarding (ii), I do not see how to translate the criterion for detecting repeated isomorphic proof steps (the *Loop Criterion*) that is used in the FOL-Decider into some similar, *general* and correct criterion that limits the depths of nested skolem functions.

Therefore, the FOL-Decider is based on a new proof strategy that I have developed for the purpose of specifying a decision procedure for FOL. I call this strategy "the $\wedge$I-minimal proof strategy in the NNF-calculus". It is based on nothing but equivalence transformation within FOL. This paper explains and proves this new strategy. Section 2 to section 4 introduce the basic relevant concepts and ideas. Section 5 explains the $\wedge$I-minimal proof strategy. Section 6 defines the *Loop Criterion* and proves that the FOL-Decider terminates in every case. The final section, section 9, proves the correctness of the implemented decision procedure based on the explanations presented in the previous sections 7 and 8.

2. Preliminaries

The following informal description of the FOL-Decider explains the principles of the implemented decision procedure, and proves its termination and correctness. The proof depends not on algorithmic details but, rather, on the basic principles of the explained decision procedure. I will restrict the presented explanations to the critical second step of the FOL-Decider. The rather trivial first step, which concerns the conversion of FOL formulas into disjunctive normal forms of FOL (FOLDNFs), is explained in my papers [Lampert (2017)] and [Lampert (2019)]. In the following, I will invoke some of the essential definitions from these papers without repeating the trivial algorithms involved or proofs of the essential theorems. [Lampert (2019)] introduces the proof strategy on which the FOL-Decider is based and illustrates it by applying it to Herbrand formulas, i.e., a fragment of FOL that is accepted to be decidable. Some familiarity with [Lampert (2019)] will facilitate a better understanding of the paper at hand but is not necessary, as I summarize the basics in the following.

[Lampert (2017)] provides some historical, philosophical and logical background concerning the use of FOLDNFs. The described algorithm for *minimizing* FOLDNFs[2], however, is irrelevant to the following. Only the algorithm for converting FOL formulas into FOLDNFs, described in section 2 of [Lampert (2017)], is of importance for the FOL-Decider. In the following, I will presume the generation of FOLDNFs from initial FOL formulas and merely provide their definition, which is based on negation normal forms (NNFs) and primary formulas.

Definition 2.1. An FOL formula is expressed in *negation normal form* (NNF) iff it contains only $\wedge$ and $\vee$ as dyadic connectives and $\neg$ appears only directly to the left of atomic propositional functions.

Definition 2.2. *Primary formulas* are defined as follows:

(1) An NNF that does not contain $\wedge$ or $\vee$ is a primary formula.
(2) NNFs that contain $\wedge$ or $\vee$ are primary formulas iff they satisfy the following conditions:
   (a) Any conjunction of n conjuncts ($n > 1$) is preceded by a sequence of existential quantifiers of minimal length 1, and all n conjuncts contain each variable of the existential quantifiers in that sequence.
   (b) Any disjunction of n disjuncts ($n > 1$) is preceded by a sequence of universal quantifiers of minimal length 1, and all n disjuncts contain each variable of the universal quantifiers in that sequence.
(3) Only NNFs that satisfy (1) or (2) are primary formulas.

Definition 2.3. An *FOLDNF* is a disjunction of length ≥ 1 of conjunctions of lengths ≥ 1 of primary formulas.

For FOLDNFs, I stipulate that all variables bound by universal quantifiers are x variables and that all variables bound by existential quantifiers are y variables. This can be easily achieved by renaming variables.

[Lampert (2019)] describes essential proof strategies for the FOL-Decider and applies them to a fragment of FOL that is known to be decidable. This fragment consists of

[2]This algorithm is also implemented as a *Mathematica* program. Like the FOL-Decider, this program can be accessed and run via the link given in footnote **??**.

formulas that are convertible into FOLDNFs with primary formulas that do not contain $\vee$ in the scope of quantifiers. The decision procedure described in [Lampert (2019)] is based on a procedure that decides, for two literals $L1$ and $L2$ of an FOLDNF, whether they are unifiable. A *unifiable pair of literals* is definable via *subformulas* ψ (cf. Definition 2.4). I will consider only unifiable pairs of literals of disjuncts D_i of an FOLDNF because an FOLDNF is refutable iff each disjunct D_i is refutable. Therefore, the decision problem for an FOLDNF reduces to the decision problem for D_i.

Definition 2.4. A *subformula* ψ is generated from a disjunct D_i of an FOLDNF as follows:

(1) Delete all literals in D_i except $L1$ and $L2$.
(2) Delete all quantifiers binding variables that do not occur in $L1$ or $L2$.
(3) Delete all occurrences of $\wedge$ and $\vee$ except the one that connects $L1$ and $L2$ in the logical hierarchy of D_i.
(4) If $L1$ and $L2$ are connected by $\vee$, replace this $\vee$ with $\wedge$.

Definition 2.5. A pair of literals $\{L1, L2\}$ from D_i is *unifiable* iff the subformula ψ generated from D_i that contains $L1$ and $L2$ is contradictory (cf. [Lampert (2019)], section 12).

A pair of literals must be unifiable to contribute to a proof of contradiction.[3]

[Lampert (2019)] describes a procedure for deciding whether a subformula ψ is refutable. The decidability of these formulas also trivially follows from the known decidability of Herbrand formulas. Thus, the unifiability of a pair of literals is decidable. Finally, [Lampert (2019)] specifies an algorithm (Algorithm 13.2) that deletes from the disjuncts D_i of an FOLDNF all literals that are not members of any unifiable pair of literals. The resulting *purged* FOLDNF is sat-equivalent to the input formula ϕ. Furthermore, I assume that the FOLDNFs are *rectified*.

Definition 2.6. *Purged* and *rectified* FOLDNFs are FOLDNFs with disjuncts D_i that satisfy the following conditions:

Purged: Every literal in D_i is a member of at least one unifiable pair of literals from D_i.

Rectified: No variable in D_i is bound by more than one quantifier.

During the purging process, literals that are not members of any unifiable pair of literals from D_i are first replaced with *sat* (for "satisfiable"). Then, as many occurrences of *sat* as possible are deleted by applying the following two *sat*-rules (cf. Algorithm 13.2 from [Lampert (2019)]):

$sat \wedge A$	$\dashv\vdash_{sat}$	A	SAT1
$sat \vee A$	$\dashv\vdash_{sat}$	sat	SAT2

Table 1: *sat*-Rules

Step 1 of the FOL-Decider results in purged and rectified FOLDNFs that are sat-equivalent to the initial input formula ϕ. The universally quantified variables of these

[3] I speak of "proofs *of* contradiction" in the case of proofs that prove that an initial formula ϕ is contradictory (= refutable). By contrast, I refer to indirect proofs (= proofs by reductio ad absurdum) that prove the negation of one of the assumptions of a deduced contradiction as "proofs *by* contradiction". The FOL-Decider is concerned not with proofs *by* contradiction but with proofs *of* contradiction.

purged and rectified FOLDNFs are x variables with indices of depth 1, and the existentially quantified variables are y variables with indices of depth 1. Thus, if D_i contains m universal quantifiers and n existential quantifiers, then D_i contains the variables x_1 through x_m and y_1 through y_n. By applying several simple auxiliary rules (cf. table 3), the FOL-Decider also simplifies the FOLDNFs in this first step. However, the extensive algorithm for minimizing FOLDNFs described in [Lampert (2017)] is not applied by default for reasons of efficiency.

Henceforth, for brevity, I will simply use the term "FOLDNFs" to refer to the purged, rectified and simplified FOLDNFs obtained as a result of step 1 of the FOL-Decider.

The refutability of certain initial formulas ϕ can already be decided in step 1 of the FOL-Decider by virtue of a simple False/sat-check. For any disjunct D_i of the resulting FOLDNF, the *DNF matrix* is generated, as defined below.

Definition 2.7. The *DNF matrix* of a formula ϕ is the scope of a prenex normal form of ϕ in disjunctive normal form (DNF).

If each disjunct of the DNF matrix contains two conjuncts, A and $\neg A$ (where A is atomic), then D_i is refutable. If D_i does not contain $\vee$ and it (and, consequently, the single disjunct of its DNF matrix) does contain a unifiable pair of literals, then it is refutable. In both cases, D_i can be deleted from the FOLDNF. If all of the D_i are refutable, then ϕ is refutable. If any disjunct of the DNF matrix of D_i does not contain any unifiable pair of literals from D_i, then ϕ is not refutable (i.e., ϕ is satisfiable). Based on this check, FOLDNFs with disjuncts D_i that do not contain $\vee$ are already decidable.

The algorithms applied in step 1 of the FOL-Decider are explained in the cited papers, where their correctness and termination are also proven. On this basis, the following (trivial) theorem can be proven:

Theorem 2.8. *The initial formula ϕ is sat-equivalent to its FOLDNF (= the result of step 1 of the FOL-Decider).*

Proof. The proof is based on the fact that all applied rules either (i) are logical equivalence rules, (ii) are sat-equivalent because only literals that are not part of a unifiable pair of literals are deleted, or (iii) rely on the described trivial False/sat-check. For details, cf. [Lampert (2019)]. □

In the following, I consider only the second step of the FOL-Decider, which decides the refutability of disjuncts D_i whose refutability is not already decided in step 1. Such a disjunct (a conjunction of n primary formulas, where $n \geq 1$) contains at least one primary formula with at least one occurrence of $\vee$ that is preceded by at least one universal quantifier. As soon as it is decided that D_i is not refutable, the FOL-Decider returns `sat`; as soon as all D_i are identified as refutable, the FOL-Decider returns `False`.

3. Basic Idea

The type of decision-making performed for D_i in step 2 of the FOL-Decider is comparable to that of algorithms for deciding whether certain numbers are finitely representable within a specific notation. The division algorithm for deciding whether a rational number has a finite representation within the decimal system may serve as a simplest illustration. If a rational number is representable by a finite decimal, then the algorithm returns that finite decimal, e.g., 0.25 in case of $\frac{1}{4}$. If it is not, then the algorithm runs in an infinite loop consisting of infinite iterations of a computational step that yields an output that dictates that the same

computation must be repeated; e.g., it repeats the computation $10 \div 3 = 3$ with remainder 1 in the case of $\frac{1}{3}$. Such a loop is detectable and can be utilized as a negative decision criterion; as soon as the computation enters such a loop, it can be concluded that no finite representation of that rational number is available in the decimal system. Consequently, the loop can be abandoned, and the computation terminates with a negative result as soon as the loop is detected.

In the case of the FOL-Decider, the question to be decided is not the equivalence of a number to a finite representation within a specific notation but rather the equivalence of a D_i to an *explicit contradiction*. This question is equivalent to the question of refutability.

Definition 3.1. An *explicit contradiction* is an NNF without universal quantifiers and with a DNF matrix in which each disjunct contains a conjunct A and a conjunct $\neg A$ (where A is atomic).

Thus, e.g., $\exists y_1 \exists y_2 (Fy_1y_2y_1 \wedge \neg Fy_1y_2y_1)$ is an explicit contradiction.

If an explicit contradiction is deducible from D_i, then the FOL-Decider returns a recipe for its deduction. If no explicit contradiction is deducible from D_i, then the steps of the proof computation along each proof path in the search tree for proofs of contradiction run, roughly speaking[4], in a detectable loop. The criterion for terminating the search for a proof on a given proof path due to the repetition of a step of the proof computation on that proof path is called the *Loop Criterion*. This criterion is defined in section 6, and it is proven to ensure the termination of the FOL-Decider in the case that no proof is found (cf. Theorem 6.18). If all proof paths for a D_i terminate without a proof of contradiction having been identified, then the FOL-Decider returns `sat`.

The general idea of proving that something is impossible by reducing endless iteration operations to "visual recursions" (detectable loops) was inspired by Wittgenstein's ideas on induction (cf. PR VIII-XIX, PG, part II.VI). The FOL-Decider applies this idea to identify unprovability by detecting loops in the proof search. The task is to define an algorithm in terms of an equivalence procedure for the application of logical operations of a correct and complete calculus such that the *Loop Criterion* can be applied to identify non-refutable formulas. As will be seen from the definition of the *Loop Criterion*, this approach still satisfies Wittgenstein's main idea of specifying a decision procedure by an algorithm such that the properties of the resulting expressions – *loop lists* in the case of the FOL-Decider – can serve as decision criteria. One might say that unlike in the axiomatic method, it is not properties that extend beyond logic that are encoded within logical formulas due to intended interpretations but rather *logical* properties that are encoded by expressions of the proper *outer form* ("a feature of certain symbols", according to TLP 4.126) that result from a purely formal, intended equivalence transformation that does not reach beyond logic.

The rationale for the proof strategy underlying the FOL-Decider is to enable the correct application of the *Loop Criterion*. The proof strategy that makes this possible is the "$\wedge$I-minimal proof strategy". Before I explain the search for proofs based on this strategy in section 5, I will introduce the calculus that is applied in the proofs of the FOL-Decider in section 4.

[4]In fact, computations along proof paths that do not result in a proof are often terminated without application of the "Loop Criterion" (cf. Example 5.31, p. 27). However, the *Loop Criterion* ensures that every proof path will terminate unless a proof is found; cf. section 6.

4. NNF-Calculus

[Lampert (2019)] introduces the NNF-calculus and proves its correctness and completeness (cf. Theorem 3.3. in [Lampert (2019)]). The correctness follows trivially from the fact that all rules are well-known derivation rules. The completeness is proven by showing that any proof within the correct and complete tree calculus can be transformed into a proof in the NNF-calculus. This section summarizes the rules of the NNF-calculus. Its rules are applied either to NNFs or for the initial generation of NNFs. Connectives such as $\rightarrow$, $\leftrightarrow$, and $|$ are eliminated using their well-known definitions immediately at the beginning of step 1 of the FOL-Decider. In the following, I omit these rules as well as the *sat*-rules listed in table 1. In addition to the sat-equivalent elimination of literals during the purging process and the rule for universal quantifier elimination (cf. table 5), the NNF-calculus comprises only well-known logical equivalence rules. In the following, I assume that $\wedge$ ties more closely than $\vee$.

$\neg\neg A$	$\dashv\vdash$	A	DN
$\neg(A \vee B)$	$\dashv\vdash$	$\neg A \wedge \neg B$	DMG$\vee$
$\neg(A \wedge B)$	$\dashv\vdash$	$\neg A \vee \neg B$	DMG$\wedge$
$A \wedge B$	$\dashv\vdash$	$B \wedge A$	COM$\wedge$
$A \vee B$	$\dashv\vdash$	$B \vee A$	COM$\vee$
$(A \wedge B) \wedge C$	$\dashv\vdash$	$A \wedge (B \wedge C)$	ASS$\wedge$
$(A \vee B) \vee C$	$\dashv\vdash$	$A \vee (B \vee C)$	ASS$\vee$
$A \wedge (B \vee C)$	$\dashv\vdash$	$A \wedge B \vee A \wedge C$	DIS1
$A \wedge B \vee C$	$\dashv\vdash$	$(A \vee C) \wedge (B \vee C)$	DIS2
A	$\dashv\vdash$	$A \wedge A$	$\wedge$I

Table 2: Equivalence Rules from Propositional Logic

$A \wedge \neg A$	$\dashv\vdash$	$\bot$	IP$\bot$0
$A \vee \neg A$	$\dashv\vdash$	$\top$	IP$\top$0
$\top \wedge A$	$\dashv\vdash$	A	IP$\top$1
$\top \vee A$	$\dashv\vdash$	$\top$	IP$\top$2
$\bot \wedge A$	$\dashv\vdash$	$\bot$	IP$\bot$1
$\bot \vee A$	$\dashv\vdash$	A	IP$\bot$2
$(A \vee B) \wedge A$	$\dashv\vdash$	A	IP1$\wedge$
$A \wedge B \vee A$	$\dashv\vdash$	A	IP1$\vee$
$A \vee A$	$\dashv\vdash$	A	$\vee$I

Table 3: Auxiliary Rules

The auxiliary rules are applied in step 1 of the FOL-Decider to simplify the FOLDNFs. Like DN, DMG$\vee$, DMG$\wedge$, DIS1 and DIS2, these rules are not applied any further in step 2 of the FOL-Decider. ASS$\wedge$, ASS$\vee$, COM$\wedge$ and COM$\vee$ are applied implicitly in the FOL-Decider by defining conjunctions and disjunctions as "orderless". Of all the rules mentioned thus far, only $\wedge$I is used in step 2 of the FOL-Decider, as will be discussed below. In step 1 of the FOL-Decider, $\wedge$I is applied only in the unproblematic direction from right to left.

$$\neg\exists\mu A(\mu) \quad \dashv\vdash \quad \forall\mu\neg A(\mu) \qquad\qquad \text{Def. } \neg\exists$$

$\neg\forall\mu A(\mu)$	$\dashv\vdash$	$\exists\mu\neg A(\mu)$	Def. $\neg\forall$
$\forall\mu\forall\nu A(\mu,\nu)$	$\dashv\vdash$	$\forall\nu\forall\mu A(\mu,\nu)$	$\forall V$
$\exists\mu\exists\nu A(\mu,\nu)$	$\dashv\vdash$	$\exists\nu\exists\mu A(\mu,\nu)$	$\exists V$
$\forall\nu(A\wedge B(\nu))$	$\dashv\vdash$	$A\wedge\forall\nu B(\nu)$	PN1
$\forall\nu(B(\nu)\wedge A)$	$\dashv\vdash$	$\forall\nu B(\nu)\wedge A$	PN2
$\forall\nu(A\vee B(\nu))$	$\dashv\vdash$	$A\vee\forall\nu B(\nu)$	PN3
$\forall\nu(B(\nu)\vee A)$	$\dashv\vdash$	$\forall\nu B(\nu)\vee A$	PN4
$\exists\nu(A\wedge B(\nu))$	$\dashv\vdash$	$A\wedge\exists\nu B(\nu)$	PN5
$\exists\nu(B(\nu)\wedge A)$	$\dashv\vdash$	$\exists\nu B(\nu)\wedge A$	PN6
$\exists\nu(A\vee B(\nu))$	$\dashv\vdash$	$A\vee\exists\nu B(\nu)$	PN7
$\exists\nu(B(\nu)\vee A)$	$\dashv\vdash$	$\exists\nu B(\nu)\vee A$	PN8
$\forall\nu(A(\nu)\wedge B(\nu))$	$\dashv\vdash$	$\forall\nu A(\nu)\wedge\forall\nu B(\nu)$	PN9
$\exists\nu(A(\nu)\vee B(\nu))$	$\dashv\vdash$	$\exists\nu A(\nu)\vee\exists\nu B(\nu)$	PN10
$\exists\mu A(\mu)$	$\dashv\vdash$	$\exists\nu A(\nu)$	SUB1
$\forall\mu A(\mu)$	$\dashv\vdash$	$\forall\nu A(\nu)$	SUB2[5]

Table 4: Equivalence Rules for Predicate Logic

Like DN, DMG$\vee$ and DMG$\wedge$, the quantifier definitions Def. $\neg\exists$ and Def. $\neg\forall$ are applied only to obtain NNFs at the beginning of step 1 of the FOL-Decider. Similar to the associative and commutative laws for $\wedge$ and $\vee$, $\forall V$ and $\exists V$ are applied implicitly in the FOL-Decider by defining sequences of similar quantifiers as orderless. The application of PN laws, however, is significant in both step 1 and step 2 of the FOL-Decider. By applying PN laws from left to right, anti-prenex normal forms can be generated. In anti-prenex normal forms, quantifiers are driven inward as far as possible through the application of PN laws. This, in turn, makes it possible to generate optimized prenex normal forms by applying PN laws from right to left. In optimized prenex normal forms, existential quantifiers are placed as far to the left of universal quantifiers as possible by means of PN laws (cf. Definition 5.8). This will be discussed below on p. 14 (cf. also [Lampert (2019)], section 6).

SUB1 and SUB2 are applied during the rectification process in step 1 of the FOL-Decider. To optimize the application of the auxiliary rules, SUB1 and SUB2 are also applied in step 1 of the FOL-Decider to achieve the opposite of rectification, namely, to use as few different variables as possible. This serves merely to simplify the FOLDNFs and is not essential for the decision procedure. In step 2 of the FOL-Decider, SUB1 and SUB2 are used only for rectification subsequent to the application of $\wedge$I (from left to right). Unlike in step 1, in step 2, the depth of the indices is incremented by 1 with each application of SUB1 or SUB2. For example, after $\wedge$I has been applied once to multiply a universally quantified expression $\forall\mu A(\mu)$, the variable μ is replaced with μ_1 in the first conjunct and with μ_2 in the second. Renaming by increasing the depth of the variables makes it possible to identify the relations of the variables, literals and pairs of literals that arise from the multiplication of conjuncts via $\wedge$I with the initial variables, literals and pairs of literals from the initial D_i in step 2 of the FOL-Decider. Indices of depths > 1 indicate *derivates*.

Definition 4.1. A *derivate* of a variable μ with an index of depth 1 is a variable that is identical to μ up to indices of depth 1. A *derivate* of a pair of literals $\{L1, L2\}$ containing variables with indices of depth 1 is a pair of literals that is identical to $\{L1, L2\}$ up to indices of depth 1.

[5]The following restriction holds for SUB1 and SUB2: ν does not occur in $A(\mu)$.

For example, x_{1_2} is a derivate of x_1, and $\{Fx_{1_2}y_{2_1}, \neg Fx_{3_{2_1}}x_{2_{1_1}}\}$ is a derivate of $\{Fx_1y_2, \neg Fx_3x_2\}$.

Aside from the *sat*-rules (cf. table 1), the following rule is the only rule in the NNF-calculus that is not a logical equivalence rule:

$$\exists\mu \ldots \forall\nu A(\mu, \nu) \quad \vdash \quad \exists\mu A(\mu, \nu/\mu) \quad \forall\text{E}$$

Table 5: Universal Quantifier Elimination

For simplicity, it is not required that $\forall\nu$ must occur directly to the right of $\exists\mu$. It is required only that $\forall\nu$ be in the scope of $\exists\mu$. Upon the application of $\forall$E, all occurrences of ν in the scope of $\forall\nu$ are replaced with μ, and $\forall\nu$ is eliminated. It is also permissible to replace ν with μ when $\exists\mu$ is a *new* existential quantifier preceding the resulting formula. I subsume this case under $\forall$E. I arbitrarily choose the variable y_0 as a new y variable and require that y_0 does not occur in the expression to the left of $\vdash$ in $\forall$E.

Step 2 of the FOL-Decider starts from the D_i; then, $\wedge$I is iteratively applied to universally quantified expressions. Each application of $\wedge$I is followed by miniscoping (by applying laws PN1-8 from left to right) and rectification (by applying SUB1 and SUB2). I denote anti-prenex expressions that result from $\wedge$I applications, miniscoping and rectification by D_i^*. In contrast to step 1 of the FOL-Decider, in step 2, neither DIS1 or DIS2 nor PN9 or PN10 is applied to generate anti-prenex normal forms D_i^* (cf. also p. 14 below). Strictly speaking, the application of $\wedge$I results in a conjunction of identical conjuncts. Henceforth, however, I subsume miniscoping and rectification under the term "$\wedge$I application". Thus, the resulting conjuncts are not identical because they vary in the indices of the variables that are bound by quantifiers occurring in the multiplied expression. Since only universally quantified expressions are multiplied in step 2 of the FOL-Decider, the resulting conjuncts vary by at least one universally quantified variable. In the case that a proof is found, the final D_i^* is also "purged*", meaning that literals that are not needed for the $\wedge$I-minimal proof are eliminated from D_i^* (cf. Definition 5.6). As long as no final D_i^* has been derived, the D_i^* are purged* only for the sake of generating optimized prenexes from *purged** anti-prenex normal forms in the steps of the proof calculation in step 2 of the FOL-Decider; cf. footnote 16 for an example. I denote optimized prenex normal forms generated from purged* D_i^* by D_i^{**}. An optimized prenex normal form D_i^{**} is optimal iff it allows an explicit contradiction to be deduced by applying $\forall$E (cf. Definition 5.22 for details). Finally, I use D_i^{***} to denote explicit contradictions deduced from D_i^{**} by applying $\forall$E (cf. figure 1).

Step 2 of the FOL-Decider computes whether explicit contradictions D_i^{***} can be deduced from the initial D_i by applying $\forall$E to optimized prenex normal forms D_i^{**}; this computation is performed by applying $\wedge$I and generating D_i^*. Strictly speaking, $\forall$E is never actually applied in the FOL-Decider. The FOL-Decider merely computes whether applications of $\forall$E subsequent to equivalence transformations *would* result in a proof of contradiction. Thus, $\forall$E is never applied to formulas that are not refutable. Moreover, in $\wedge$I-minimal proofs consistent with the recipe returned by the FOL-Decider, $\forall$E will only be applied to formulas identified as refutable if doing so will ultimately allow explicit contradictions to be deduced. Thus, the decision-making of the FOL-Decider is based on nothing but equivalence transformations with regard to the property of refutability, which is the property in question.

There is no need for a rule for existential quantifier elimination in the NNF-calculus. Unlike in the tree calculus (tableau calculus), quantified expressions are not decomposed

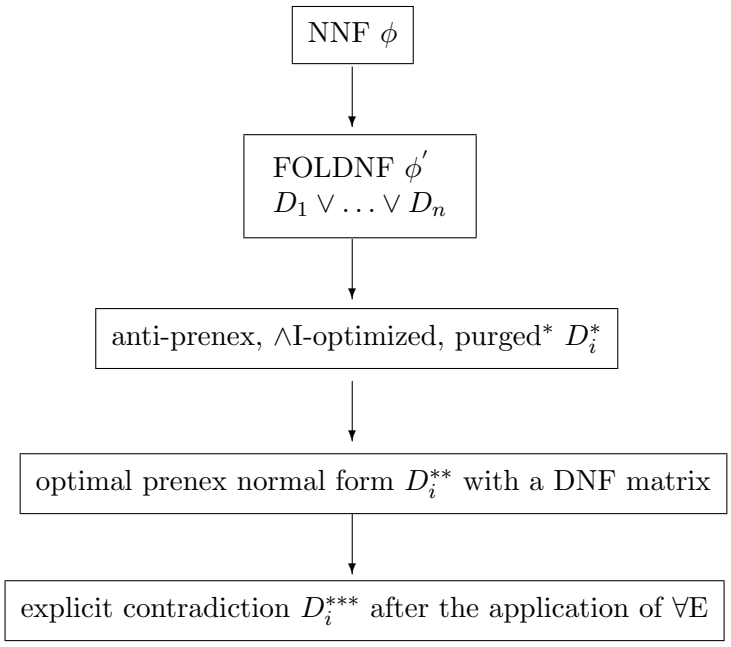

Figure 1: Steps of a proof in the NNF-calculus

multiple times. Instead, in the proofs of the NNF-calculus, universally quantified expressions $\forall\mu A(\mu)$ are multiplied by applying $\wedge$I. Consequently, the universal quantifier $\forall\mu$ and its variable μ are multiplied. After rectification and the subsequent suitable generation of an optimized prenex normal form, the multiplied x variable can then be replaced with different y variables in literals stemming from different conjuncts by applying $\forall$E (cf. Example 4.1, [Lampert (2019)], p. 7, and the discussion on p. 20 below). I call the process of computing and applying $\wedge$I in step 2 of the FOL-Decider "$\wedge$I-optimization".

In the framework of the NNF-calculus, the decision problem for D_i consists of defining a criterion for the $\wedge$I applications performed to achieve multiplications of universally quantified expressions that are necessary and sufficient to find a proof of contradiction via *unification* by applying $\forall$E. In contrast to Definition 2.3 of [Lampert (2019)], which defines unification for *single* pairs of literals of a subformula ψ, the following Definition 4.2 defines unification for a *set* of unifiable pairs of literals.[6]

Definition 4.2. *Unification* is the result of replacing (universally quantified) x variables such that identical positions[7] in a unifiable pair of literals from D_i^* are occupied by identical (existentially quantified) y variables. A *set* of unifiable pairs of literals from D_i^* is *unifiable* if identical y variables can be made to occupy identical positions in all pairs of literals in that set by applying a logically valid substitution procedure. A *set* of unifiable pairs of

[6]In contrast to Definition 2.3 of [Lampert (2019)], which implies that *unification* is a logically valid process in the case of a *single* pair of literals, Definition 4.2 does not presuppose that the unification of *all* pairs of literals is logically valid. It merely presupposes, in accordance with Definition 2.3 of [Lampert (2019)], that the unification of each *single* pair of literals from a set of unifiable pairs of literals in the corresponding subformula ψ is logically valid. In contrast to the *unification* of a set of pairs of literals, the *unifiability* of such a set presupposes a logically valid procedure according to Definition 4.2. Proofs of contradiction depend on the unifiability of sets of pairs of literals.

[7]I identify positions in pairs of literals with respect to the locations of the arguments of the corresponding predicates of the literals. Thus, x_1 and y_1 occur in identical positions in $\{Fx_1, \neg Fy_1\}$.

literals is *unified* if all identical positions in all pairs of literals are occupied by identical y variables.

In proofs within the NNF-calculus, identical positions in literals are unified by applying ∀E. Step 2 of the FOL-Decider computes, prior to any application of ∀E, whether (minimal) sets of pairs of literals that would suffice to enable the deduction of explicit contradictions D_i^{***} are *unifiable* within the (correct and complete) NNF-calculus through the generation of D_i^* and D_i^{**} starting from D_i.

5. Searching for ∧I-minimal Proofs

5.1. **∧I-minimal Proofs.** In step 2 of the FOL-Decider, the only rule applied that increases the complexity is ∧I in the direction from left to right. ∧I is applied to replace a universally quantified x variable in different conjuncts with different existentially quantified y variables for the purpose of unification. In step 2 of the FOL-Decider, different proof paths are generated to enable a systematic search for ∧I-minimal proofs. Each proof path is determined by a specific combination of ∧I applications to unify pairs of literals for the deduction of explicit contradictions via the application of ∀E. According to the completeness theorem of the NNF-calculus (Theorem 3.3 in [Lampert (2019)]), a proof based on ∧I applications in the NNF-calculus exists if D_i is refutable. The search for proofs in the NNF-calculus can be reduced to a search for ∧I-minimal proofs.

Definition 5.1. A proof of contradiction in the NNF-calculus is *∧I-minimal* iff each application of ∧I is necessary. A ∧I application in a proof of contradiction for D_i is *necessary* iff each conjunct in the final D_i^* resulting from that ∧I application is necessary. A *conjunct* in the final D_i^* is *necessary* iff eliminating it causes no explicit contradiction D_i^{***} to be deduced any longer.

One can decide whether the condition for a necessary conjunct in the final D_i^* is satisfied as follows. Let C be a conjunct in the final D_i^* that results from a ∧I application. Let `lits` denote the literals in D_i^{***} that are generated from the literals in C by replacing x variables with y variables. Let M be the DNF matrix from D_i^{***}, and let M' be the matrix that is obtained from M if one eliminates the literals `lits` in M. Then, C in D_i^* is necessary for the proof of contradiction iff it is not the case that each disjunct of M' contains two conjuncts, A and $\neg A$; cf. Example 5.2 in [Lampert (2019)].

Theorem 5.2. *If D_i is refutable, then a ∧I-minimal proof of D_i exists.*

Proof. From the completeness of the NNF-calculus (cf. Theorem 3.3 in [Lampert (2017)]), it follows that a proof in the NNF-calculus exists if D_i is refutable. Any proof of D_i in the NNF-calculus, however, can be reduced to a ∧I-minimal proof by eliminating unnecessary ∧I applications. Therefore, a ∧I-minimal proof of D_i exists if D_i is refutable. □

The FOL-Decider calculates only necessary and sufficient conditions for ∧I-minimal proofs. Either it terminates the search for a proof on a particular proof path, if a necessary condition for a ∧I-minimal proof is not satisfied, or it returns a recipe for either (i) a ∧I-minimal proof (normal case) or (ii) an overdetermined ∧I-minimal proof that contains a ∧I-minimal proof (abnormal case). The latter may arise due to insufficient optimization of the ∧I-minimal proof strategy such that the proof search is not reduced to ∧I-minimal proofs only. For the question of decidability, it is sufficient to reduce the search space to a

finite one that contains all ∧I-minimal proofs and to identify either a ∧I-minimal proof or an overdetermined ∧I-minimal proof within this finite search space (cf. section 8).

Step 2 of the FOL-Decider generates anti-prenex disjuncts D_i^* with universally quantified expressions that have been multiplied due to ∧I applications. For simplicity, I also subsume D_i under expressions of the D_i^* type. From this perspective, D_i is the result of applying ∧I 0 times. The FOL-Decider does not, in fact, generate D_i^{**} or D_i^{***}. In the case that a proof of contradiction for D_i is found, it returns (i) the final purged*[8] anti-prenex normal form D_i^*, (ii) the final minimal set `L` of pairs of literals from D_i^* that constitute the identified ∧I-minimal proof of contradiction, (iii) a final substitution list σ that specifies how the x variables should be replaced with y variables in `L` to unify all literals from `L`, and (iv) the final prenex $\wp$ of D_i^{**} that allows ∀E to be applied to replace x variables with y variables as prescribed by σ and, consequently, to generate D_i^{***}.

5.2. **Parameters of a ∧I-minimal Proof Search: `L`, $\wp$, and `sub`.** Step 2 of the FOL-Decider calculates the ∧I applications that are necessary in the search for a ∧I-minimal proof. Each proof step on a proof path consists of an application of ∧I. Its calculation depends on the following parameters:

(1) a minimal set `L` of unifiable pairs of literals,
(2) an optimized prenex $\wp$,
(3) a specification `sub` of substitutions that unify the single pairs of literals in `L`, and
(4) a loop list.

From the substitution specification `sub`, a substitution list σ can be generated that specifies, for each x variable ν, the y variables with which ν must be replaced to unify all pairs of literals in `L`.

Remark 5.3. If `sub` specifies that an x variable `xvar` must be replaced with several y variables `yvars`, this means that instances of `xvar` in different literals must be replaced with different y variables from `yvars` to unify all pairs of literals in `L`. In accordance with the ∧I-minimal proof strategy, this unification can be achieved in a logically valid way only through ∧I applications.

The first three parameters specify the substitutions of the x variables and, consequently, the applications of ∧I. Alternative specifications of these parameters result in alternative proof paths. This section explains these three parameters in detail, whereas section 6 discusses the loop list.

Definition 5.4. A set `L` of unifiable pairs of literals from D_i^* is *minimal* iff (i) unification of these pairs of literals from D_i^* is sufficient to satisfy the condition that each disjunct of the DNF matrix of D_i^* contains two contradictory literals, A and $\neg A$, and (ii) said condition is no longer satisfied if any one of these pairs of literals is eliminated from `L`.

Thus, the unification of `L` is *minimally sufficient* to satisfy the condition that each disjunct of the DNF matrix contains an explicit contradiction.

Definition 5.4 does not presuppose that it is possible to unify all pairs of literals from `L` in a logically valid way by deducing an explicit contradiction D_i^{***} from D_i^*. It presupposes only that each *single* pair of literals can be unified in a logically valid way by deducing an explicit contradiction from the corresponding subformula ψ. As will be explained in

[8] Cf. p. 9 and Definition 5.6.

more detail later in this section, the parameters L, $\wp$ and sub are defined such that one can conclude the (logically valid – cf. Definition 4.2) unifiability of *all* pairs of literals from L *as a whole* only if no x variable in L must be replaced with more than one y variable in order to unify all pairs of literals in L according to sub. If the unification of L requires any x variable to be replaced with more than one y variable according to sub, then $\wedge$I must be applied. Thus, within the framework of the $\wedge$I-minimal proof strategy, whether all pairs of literals of a minimal set L are unifiable can be decided only by applying $\wedge$I. This $\wedge$I application may cause it to be necessary to extend L in order to satisfy condition (i) specified in Definition 5.4.

Remark 5.5. The unification of each pair of literals in a minimal set L of unifiable pairs of literals is a *necessary* condition for a $\wedge$I-minimal proof on a given proof path. In the case that no x variable must be replaced with more than one y variable according to sub, I call sub "unambiguous". As will be shown below on p. 19, the unambiguity of sub is a *sufficient* condition for the identification of a $\wedge$I-minimal proof.

By referring to L, one can define purged* anti-prenex normal forms as follows:

Definition 5.6. An anti-prenex normal form D_i^* is *purged** iff D_i^* contains only literals from L.

The *Fundamental Principle* of the $\wedge$I-minimal proof strategy underlying Definition 5.1 can be specified with reference to L. To do so, I first define "selected conjuncts". In this definition, I presume that conjunctions of length m ($m \geq 1$) contain m conjuncts.

Definition 5.7. *Selected conjuncts* are (i) conjuncts from D_i^* that contain literals in L and (ii) all conjuncts resulting from a $\wedge$I application.

This definition relates to any stage of the proof. It applies to any modification of D_i^* and L.

The *Fundamental Principle* can be defined in relation to selected conjuncts (Definition 5.7) and necessary conjuncts (Definition 5.1) as follows:

Fundamental Principle (FP). *Any once-selected conjunct must be necessary for a $\wedge$I-minimal proof.*

Starting from the conjuncts in D_i prior to any application of $\wedge$I, FP necessitates that once they have been selected, conjuncts will never be "dropped" in the proof search. In the course of further $\wedge$I applications on a proof path, the selected conjuncts can only be further multiplied. Every step on a proof path involves the replacement of one selected conjunct with n ($n > 1$) conjuncts. Consequently, L must always contain at least one literal from each conjunct that results from an application of $\wedge$I. If this is not the case, the proof path can be terminated because it will not result in a $\wedge$I-minimal proof (cf. Example 5.28, p. 25 below). If a proof is found, it is based on all selected conjuncts and, therefore, all $\wedge$I applications along the proof path.

To find a $\wedge$I-minimal proof if it exists, the FOL-Decider generates *all* minimal sets L. Different sets L generated from D_i^* correspond to different alternative proof paths. $\wedge$I-minimal proofs are restricted to *minimal* sets L to keep the required number of substitutions low. Consequently, the number of $\wedge$I applications needed to replace x variables with different y variables is kept low.

Optimized prenexes $\wp$ are prenexes with universal quantifiers as far to the right of existential quantifiers as possible.

Definition 5.8. A prenex $\wp$ of a prenex normal form D_i^{**} that is generated from the anti-prenex normal form D_i^* is *optimized* iff the universal quantifiers are in the scope of the existential quantifiers to the greatest possible extent through the application of PN1-8.

Algorithm 10.1 in [Lampert (2019)] defines the process of generating all possible optimized prenex normal forms from a subformula ψ. In the case of anti-prenex normal forms D_i^* that also contain $\vee$, this algorithm must be extended such that PN3-4 and PN7-8 are also considered. Because this extension is trivial and adds nothing fundamentally new, I abstain here from presenting the full algorithm for generating all optimized prenexes from a given anti-prenex normal form D_i^*. In fact, the FOL-Decider applies a procedure that is more effective than Algorithm 10.1 in [Lampert (2019)]. In the FOL-Decider, prenexes $\wp$ are represented as lists of x and y variables. Optimized prenexes are generated combinatorially from different sequences of x and y variables. In doing so, the algorithm refers to *purged** anti-prenex normal forms D_i^* that contain only literals from `L`. The key to the $\wedge$I-minimal proof strategy is that generating prenex normal forms from *anti-prenex* normal forms makes it possible to generate all and only *optimized* prenexes with universal quantifiers that are in the scope of the existential quantifiers to the maximal extent. This increases the possibility to apply $\forall$E for unification without needing to apply $\wedge$I (cf. section 6 in [Lampert (2019)] and p. 19 below). Alternative possibilities for generating optimized prenexes from anti-prenex normal forms give rise to alternative proof paths.

Step 1 of the FOL-Decider increases the number of quantifiers through the application of PN9 and PN10 and through the conversion of the scope of universal quantifiers into conjunctive normal form (CNF) and the scope of existential quantifiers into DNF. This is done so that the scope of quantifiers is minimized to the maximal extent when the PN laws are applied. For simplicity, these strategies are not applied in step 2 of the FOL-Decider when generating anti-prenex normal forms.[9] Step 2 of the FOL-Decider increases the number of quantifiers and the number of variables they bind only as a result of $\wedge$I applications. Therefore, the strategy for minimizing the number of necessary $\wedge$I applications in a proof of contradiction is not absolute but instead depends on D_i^* and D_i^{**}, in which the number of quantifiers does not change during the process of miniscoping or prenexing.[10]

Henceforth, I use `L` to refer to *minimal* sets of pairs of literals and $\wp$ to refer to *optimized* prenexes.

`sub` specifies substitutions to be made in order to unify pairs of literals in `L`. Two cases must be distinguished when specifying these substitutions. In *case 1*, a single pair of literals unambiguously determines a specific y variable that must replace an x variable in a certain position to unify that pair of literals. In *case 2*, a single pair of literals merely determines that two x variables, occurring in identical positions in $L1$ and $L2$, must be replaced with

[9]The number of $\wedge$I applications can also be decreased by minimizing the scope of existential quantifiers above conjunctions to the extent that this is sat-equivalent (cf. the discussion of $\exists$M optimization in section 7 of [Lampert (2019)]). For simplicity, however, the FOL-Decider makes use of $\exists$M optimization only to identify unifiable pairs of literals. $\exists$M optimization is applied neither when generating FOLDNFs in step 1 nor when generating anti-prenex normal forms D_i^* in step 2. Nevertheless, $\exists$M optimization is an *optional* tool in step 1 of the FOL-Decider (cf. also footnote 9 in [Lampert (2019)]).

[10]The number of $\wedge$I applications on a proof path is also related to the extent to which the FOLDNFs and D_i^* are simplified. The number of $\wedge$I applications could also, of course, be minimized via extensive minimization procedures such as those described in [Lampert (2017)]. However, only rare use is made of the simplification of FOLDNFs in step 1 of the FOL-Decider. In step 2, the D_i^* are merely simplified by deleting literals that are not included in `L` (purging) to generate optimized prenexes $\wp$ or to return the final D_i^*.

the same y variable in those positions, without dictating any specific y variable for that purpose. *Case 1* is illustrated by $\{Fx_1, \neg Fy_1\}$; *case 2*, by $\{Fx_1, \neg Fx_2\}$. In *case 2*, the substitutions of the x variables may be determined by substitutions of x variables in *other* unifiable pairs of literals in `L` (*case 2.1*), or they may not (*case 2.2*). In *case 2.1*, multiple alternative possibilities for replacing x variables with y variables for unification may arise.

Example 5.9. Let D_i be given as follows:

$$\exists y_1 G y_1 y_1 \wedge \forall x_1 (Fx_1 \vee \forall x_3 (\forall x_4 H x_1 x_3 x_4 \vee Jx_1 x_3)) \wedge$$
$$\forall x_2 \exists y_2 (\neg Fy_2 \wedge \forall x_5 (\neg G x_2 x_5 \vee \neg H x_2 y_2 x_5) \wedge \neg J x_2 y_2) \tag{5.1}$$

There is exactly one minimal set `L` of unifiable pairs of literals for (5.1):

$$\{\{Fx_1, \neg Fy_2\}, \tag{5.2}$$
$$\{Gy_1 y_1, \neg G x_2 x_5\}, \tag{5.3}$$
$$\{Hx_1 x_3 x_4, \neg H x_2 y_2 x_5\}, \tag{5.4}$$
$$\{Jx_1 x_3, \neg J x_2 y_2\}\} \tag{5.5}$$

There is also only one optimized prenex $\wp$ for (5.1), which is represented by a list of x and y variables, according to the FOL-Decider:

$$\{y_1, x_2, y_2, x_1, x_3, x_4, x_5\} \tag{5.6}$$

In (5.2), x_1 must be replaced with y_2; in (5.3), x_2 must be replaced with y_1, and x_5 must be replaced with y_1; in (5.4), x_1 and x_2 must both be replaced with the same y variable, x_3 must be replaced with y_2, and x_4 and x_5 must both be replaced with the same y variable; and in (5.5), x_1 and x_2 must both be replaced with the same y variable, and x_3 must be replaced with y_2. In contrast to x_3, x_4 and x_5, at least one of the x variables x_1 and x_2 must replaced with more than one y variable to unify all pairs of literals in (5.2) to (5.5). Their possible substitutions in identical positions in (5.4) and (5.5) are determined by the specifications of their substitutions in (5.2) and (5.3). Since x_1 must be replaced with y_2 in (5.2) and x_2 must be replaced with y_1 in (5.3) (*case 1*), four possible substitutions for x_1 and x_2 in (5.4) and (5.5) arise through combinatorial means (*case 2.1*):

(1) x_1 and x_2 may be replaced with y_1 in both (5.4) and (5.5),
(2) x_1 and x_2 may be replaced with y_2 in both (5.4) and (5.5),
(3) x_1 and x_2 may be replaced with y_1 in (5.4) and with y_2 in (5.5), or
(4) x_1 and x_2 may be replaced with y_2 in (5.4) and with y_1 in (5.5).

(1) necessitates the replacement of x_1 with y_1 in addition to its replacement with y_2 in (5.2) and (2) necessitates the replacement of x_2 with y_2 in addition to its replacement with y_1 in (5.3). In both cases, however, only x_1 (in (1)) or only x_2 (in (2)) must be replaced with more than one y variable to unify the pairs of literals in `L`. By contrast, (3) and (4) necessitate the replacement of both x_1 and x_2 with both y_1 and y_2. The FOL-Decider realizes *all* of the combinatorially possible substitutions for *case 2.1* along different proof paths. Thus, specifications `sub` such as (3) and (4) that result in more substitutions than other possible specifications `sub` such as (1) and (2) are not excluded. The reason for this is that it cannot be guaranteed that an initial specification `sub` that requires more applications of $\wedge$I may not result in a `sub` that requires fewer applications of $\wedge$I at a later stage along the proof

path.[11] If all combinatorially possible substitutions for *case 2.1* are generated in alternative specifications `sub`, then no ∧I-minimal proof will be excluded. It might be possible to define

[11]This is due to minimal extensions, as necessitated by ∧I applications, of minimal sets `L` of literals. The following example illustrates such a case.

Example 5.10. Let D_i be given as follows:

$$\forall x_1 \exists y_1 F x_1 y_1 \wedge \forall x_2 \forall x_5 (\forall x_7 (F x_5 x_7 \vee \neg F x_2 x_7) \vee \neg F x_5 x_2) \wedge$$
$$\forall x_3 (\exists y_2 (F x_3 y_2 \wedge G y_2) \vee G x_3) \wedge \forall x_4 (\forall x_6 (\neg F x_4 x_6 \vee \neg G x_6) \vee \neg G x_4) \tag{5.7}$$

One of the several initial sets `L` of pairs of literals is as follows:

$$\{\{G x_3, \neg G x_4\}, \tag{5.8}$$
$$\{G x_3, \neg G x_6\}, \tag{5.9}$$
$$\{G y_2, \neg G x_4\}, \tag{5.10}$$
$$\{G y_2, \neg G x_6\}, \tag{5.11}$$
$$\{F x_1 y_1, \neg F x_4 x_6\}\} \tag{5.12}$$

This selection of pairs of literals results in a ∧I-minimal proof only if one specifies that x_3 and x_4 are both to be replaced with y_2 in (5.8) and that x_3 and x_6 are both to be replaced with y_1 in (5.9). This determines that both x_3 and x_6 must be replaced with y_1 and y_2. This would be avoided if it were to be specified that x_3 and x_6 were both to be replaced with y_2 in (5.9). However, this would cause the ∧I-minimal proof, in which ∧I is applied to replace both x_6 and x_3 with y_1 and y_2 (or suitable derivates of y_1 and y_2), not to be found. By contrast, if `sub` initially specifies that both x_3 and x_6 must be replaced with y_1 and y_2, then a ∧I-minimal proof is found with the following D_i^*:

$$\forall x_1 \exists y_1 F x_1 y_1 \wedge \forall x_2 \forall x_5 (\forall x_7 (F x_5 x_7 \vee \neg F x_2 x_7) \vee \neg F x_5 x_2) \wedge$$
$$\forall x_{3_1} (\exists y_{2_1} (F x_{3_1} y_{2_1} \wedge G y_{2_1}) \vee G x_{3_1}) \wedge (\exists y_{2_2} G y_{2_2} \vee \forall x_{3_2} G x_{3_2}) \wedge \tag{5.13}$$
$$\forall x_{4_1} (\forall x_{6_1} (\neg F x_{4_1} x_{6_1} \vee \neg G x_{6_1}) \vee \neg G x_{4_1}) \wedge \forall x_{4_2} (\forall x_{6_2} (\neg F x_{4_2} x_{6_2} \vee \neg G x_{6_2}) \vee \neg G x_{4_2})$$

In the course of applying ∧I to multiply $\forall x_3$, $\exists y_2$ is also multiplied because $\exists y_2$ is in the scope of $\forall x_3$ in (5.7). On the proof path that results in the proof of contradiction, extending `L` determines that $F x_{3_2} y_{2_2}$ is not a member of a pair of literals contained in `L`. Thus, $F x_{3_2} y_{2_2}$ is deleted from the final D_i^* (5.13). This, in turn, determines that $\exists y_{2_2}$ is not in the scope of $\forall x_{3_2}$ as a result of the scope minimization leading to (5.13). This makes it possible to generate the optimized (and optimal) prenex given in (5.25), with y_{2_2} to the left of x_{3_2}, which allows x_{3_2} to be replaced with y_{2_2} by applying ∀E. As a consequence of multiplying $\forall x_6$, $\forall x_4$ is also multiplied (cf. p. 20). In contrast to x_{6_1} and x_{6_2}, however, x_{4_1} and x_{4_2} are not replaced with different y variables in the final ∧I-minimal proof. The minimal set `L` of pairs of literals given in (5.8) - (5.12) must be extended due to the sequence of ∧I applications. This results in the following final `L`:

$$\{\{G x_{3_2}, \neg G x_{4_1}\}, \tag{5.14}$$
$$\{G x_{3_1}, \neg G x_{6_1}\}, \tag{5.15}$$
$$\{G y_{2_2}, \neg G x_{4_1}\}, \tag{5.16}$$
$$\{G y_{2_2}, \neg G x_{6_2}\}, \tag{5.17}$$
$$\{G x_{3_2}, \neg G x_{4_2}\}, \tag{5.18}$$
$$\{G y_{2_1}, \neg G x_{6_2}\}, \tag{5.19}$$
$$\{F x_1 y_1, \neg F x_{4_1} x_{6_1}\}, \tag{5.20}$$
$$\{F x_5 x_7, \neg F x_{4_2} x_{6_2}\}, \tag{5.21}$$
$$\{F x_1 y_1, \neg F x_5 x_2\}, \tag{5.22}$$
$$\{F x_{3_1} y_{2_1}, \neg F x_2 x_7\}\} \tag{5.23}$$

The following substitution list σ specifies how to replace x variables with y variables to unify all pairs of literals in (5.14) - (5.23):

$$\{\{x_1, y_{2_2}\}, \{x_2, y_1\}, \{x_5, y_{2_2}\}, \{x_7, y_{2_1}\}, \{x_{3_1}, y_1\}, \{x_{3_2}, y_{2_2}\},$$
$$\{x_{4_1}, y_{2_2}\}, \{x_{4_2}, y_{2_2}\}, \{x_{6_1}, y_1\}, \{x_{6_2}, y_{2_1}\}\} \tag{5.24}$$

criteria for restricting the number of alternative specifications `sub` without excluding any ∧I-minimal proofs. However, no such criteria are defined in the FOL-Decider for now. This is one of the reasons why an enormous number of alternative proof paths are generated in step 2 of the FOL-Decider.

To specify `sub` further, let us introduce the concept of *xx positions* of a pair of unifiable literals. Roughly speaking, identical positions in two literals forming a unifiable pair $\{L1, L2\}$ are *xx positions* iff (i) they are occupied by x variables and (ii) the occurrence of x and y variables in certain positions in $L1$ and $L2$ does not determine specific y variables that must replace x variables occurring in identical positions for the sake of unification. Thus, *xx* positions are relevant to *case 2.1* and *case 2.2*. To define *xx* positions more precisely and algorithmically, let us recall Definitions 7.3 and 7.5 of [Lampert (2019)], which define *xx* lists as well as both *xy* and *yx* pairs:

Definition 7.3: The *xx lists* of a pair of connected literals $\{L1, L2\}$ are generated as follows:

(1) If an x variable ν_1 and an x variable ν_2 occur in identical positions in L1 and L2, then ν_1 and ν_2 are members of the same *xx* list.

(2) If the x variables ν_1 and ν_2 are members of the same *xx* list and if ν_2 and the x variable ν_3 are also members of the same *xx* list, then ν_1, ν_2 and ν_3 are all members of the same *xx* list.

(3) (1) and (2) are the only conditions that determine members of the same *xx* list.

Definition 7.5: An *xy pair* is an ordered list $\{xvar, yvar\}$ consisting of an x variable $xvar$ and a y variable $yvar$, where $xvar$ occurs in L1 in a position n and $yvar$ occurs in L2 in the same position n. A *yx pair* is an ordered list $\{yvar, xvar\}$ consisting of a y variable $yvar$ and an x variable $xvar$, where $yvar$ occurs in L1 in a position n and $xvar$ occurs in L2 in the same position n.

Based on these definitions, *xx* positions are defined as follows:

Definition 5.11. Let all x variables of $L1$ be subscripted with -1, and let all x variables of $L2$ be subscripted with -2. Then, identical positions in $L1$ and $L2$ are *xx positions* iff (i) they are occupied by x variables and (ii) these x variables are members of *xx* lists, which contain no x variables that are members of an *xy* or *yx* pair.

The condition specified in the first sentence of the above definition is important because it cannot be presupposed that the same x variable is to be replaced with the same y variable in both $L1$ and $L2$ (cf. Remark 5.13).

Note that neither other pairs of literals in the same `L` nor the order of the quantifiers in D_i^* that bind variables from $\{L1, L2\}$ are considered in the definition of *xx* positions.

Example 5.12. In $\{Fx_1x_1, \neg Fx_2y_1\}$, x_1 and x_2 do not occur in *xx* positions because x_1 must be replaced with y_1 in position 2 of Fx_1x_1 for unification. Therefore, x_1 must also be replaced with y_1 in position 1 in the same literal. Consequently, the first position in $L1$

The following prenex $\wp$ allows for these substitutions:

$$\{y_{2_2}, x_1, y_1, x_{3_1}, y_{2_1}, x_7, x_5, x_2, x_{6_1}, x_{4_1}, x_{6_2}, x_{4_2}, x_{3_2}\} \tag{5.25}$$

If one generates the prenex normal form D_i^{**} with the prenex given in (5.25) from the anti-prenex normal form given in (5.13), then all substitutions in (5.24) can be realized by applying ∀E.

and the first position in $L2$ are not xx positions. The occupation of identical positions in $L1$ and $L2$ by x variables is a necessary but not a sufficient condition for these positions to be xx positions.

xx positions can be occupied by the same x variable, as in the case of, e.g., $\{Fx_1, \neg Fx_1\}$.

Example 5.13. It might well be that the same x variable is to be replaced with different y variables in $L1$ and $L2$. For example, in $\{Fx_1y_1x_3, \neg Fx_2x_1y_1\}$, x_1 must be replaced only with y_1 in $L2$; in $L1$, however, x_1 occurs in an xx position. Therefore, the pair of literals does not dictate that x_1 must be replaced with y_1 in the first position of $L1$. Thus, the FOL-Decider proves that the formula $\forall x_1 \exists y_1 (\forall x_3 Fx_1y_1x_3 \wedge \forall x_2 \neg Fx_2x_1y_1)$ is refutable by means of a single application of $\wedge$I to replace x_{1_1} in $Fx_{1_1}y_{1_1}x_{3_1}$ with y_0 and x_{1_2} in $\neg Fx_{2_2}x_{1_2}y_{1_2}$ with y_{1_1} in the resulting D_i^{**}.[12]

Definition 5.14. A *specification* `sub` *of substitutions* specifies a set of substitutions of x variables in all positions in literals in `L` such that all single pairs of literals are unified.

The FOL-Decider combines `L` and `sub` into a list denoted by `sub/L`, in which each pair of literals from `L` is preceded by a list that specifies how its x variables are to be substituted according to `sub`. Each `sub` for a single pair of literals is a list of three lists. The first list specifies which y variables must replace x variables due to unification requirements for that single pair of literals (cf. *case 1* on p. 14). The second and the third list concern xx positions. The second list specifies how x variables must be replaced with y variables as dictated by substitutions in other pairs of literals in `L` (cf. *case 2.1* on p. 14). The third list specifies x variables without specific substitution requirements; it simply specifies which of them must be replaced with the same y variable (cf. *case 2.2* on p. 15). Each of these three lists concerns the substitution requirements for x variables in certain positions in the corresponding pair of literals. These lists and the corresponding pair of literals together dictate how the variable in each position that is occupied by an x variable must be substituted. The generation of `sub/L` and its modifications and extensions all play crucial roles in the FOL-Decider.

Example 5.15. The `sub/L` for (5.2) - (5.5) from Example 5.9 that corresponds to specification (1) for the xx positions of x_1 and x_2 in (5.4) and (5.5) (cf. p. 15) is as follows:

$$\{\{\{\{\{x_1, y_2\}\}, \{\}, \{\}\}, \{Fx_1, \neg Fy_2\}\}, \tag{5.28}$$

$$\{\{\{\{x_2, y_1\}, \{x_5, y_1\}\}, \{\}, \{\}\}, \{Gy_1y_1, \neg Gx_2x_5\}\}, \tag{5.29}$$

$$\{\{\{\{x_3, y_2\}\}, \{\{x_1, x_2, y_1\}, \{x_4, x_5, y_1\}, \{\}\}, \{Hx_1x_3x_4, \neg Hx_2y_2x_5\}\}, \tag{5.30}$$

$$\{\{\{\{x_3, y_2\}\}, \{\{x_1, x_2, y_1\}\}, \{\}\}, \{Jx_1x_3, \neg Jx_2y_2\}\}\} \tag{5.31}$$

Remark 5.16. The substitutions of x variables in xx positions that are specified in `sub` depend on the pairs of literals from `L`. As discussed on p. 19f., the replacement of x variables

[12]The resulting D_i^{**} is

$$\forall x_{1_1} \exists y_{1_1} \forall x_{1_2} \exists y_{1_2} \forall x_{3_1} \forall x_{2_2} (Fx_{1_1}y_{1_1}x_{3_1} \wedge \neg Fx_{2_2}x_{1_2}y_{1_2}) \tag{5.26}$$

This D_i^{**} is generated from the following purged* D_i^*:

$$\forall x_{1_1} \exists y_{1_1} \forall x_{3_1} Fx_{1_1}y_{1_1}x_{3_1} \wedge \forall x_{1_2} \exists y_{1_2} \forall x_{2_2} \neg Fx_{2_2}x_{1_2}y_{1_2} \tag{5.27}$$

Note that $\exists$M (cf. p. 10 in [Lampert (2019)]) is not applied by default in step 1 of the FOL-Decider (cf. footnote 9 above and footnote 9 in [Lampert (2019)]). Therefore, the FOL-Decider proves that $\forall x_1 \exists y_1 (\forall x_3 Fx_1y_1x_3 \wedge \forall x_2 \neg Fx_2x_1y_1)$ is refutable by means of a single application of $\wedge$I.

with y_0 to satisfy condition $\mathtt{P}_\wp$ additionally depends on optimized prenexes $\wp$ and is also considered in `sub`.

The FOL-Decider generates all possible alternative specifications `sub` for the unification of the pairs of literals in `L`.

Definition 5.17. A *substitution list* σ for a set `L` of unifiable pairs of literals is a list consisting of lists that specify, for each x variable from `L`, which y variables must replace it according to `sub` for the unification of all pairs of literals in `L`.

Remark 5.18. The FOL-Decider generates substitution lists σ from `sub`.

Example 5.19. The substitution list σ for the `sub` given in (5.28) to (5.31) is

$$\sigma: \quad \{\{x_1, y_1.y_2\}, \{x_2, y_1\}, \{x_3, y_1\}, \{x_4, y_1\}, \{x_5, y_1\}\} \tag{5.32}$$

Strictly speaking, the FOL-Decider differentiates between substitution lists, which contain only x variables that are to be replaced with at least one y variable according to `sub`, and final substitution lists, which contain all x variables from `sub`. This distinction concerns x variables ν for which no specific y variables are specified in `sub` to replace them (cf. *case 2.2*, p. 15). Final substitution lists include lists of the form $\{\nu, y_0\}$ for these x variables. However, the final substitution lists are a rather insignificant detail of the FOL-Decider; they are relevant only in the case that a proof is found. Thus, I abstain from distinguishing between these two kinds of substitution lists in the following. Unless the substitution lists σ at hand are those returned by the FOL-Decider in the case that a $\wedge$I-minimal proof is found, I refer to substitution lists σ in the sense of the first meaning.[13]

Definition 5.20. A substitution list σ is *unambiguous* iff each partial list in σ is of length 2 (i.e., contains exactly one y variable).

Remark 5.21. σ is unambiguous iff `sub` is unambiguous (cf. p. 13).

As soon as σ is unambiguous apart from the introduction of y_0 into `sub` to satisfy $\mathtt{P}_\wp$, `sub` must satisfy the *Prenex Principle* $\mathtt{P}_\wp$.

Prenex Principle ($\mathtt{P}_\wp$). *Each x variable ν of an optimized prenex $\wp$ appears to the right of at least one y variable that is to be substituted for ν according to the specification* `sub` *of substitutions.*

$\mathtt{P}_\wp$ ensures that an explicit contradiction D_i^{***} can be deduced from D_i given that σ is unambiguous and $\mathtt{P}_\wp$ is satisfied. In this case, the anti-prenex normal form D_i^* that is deduced from D_i via $\wedge$I applications can be purged* and converted into a prenex normal form D_i^{**} with an optimized prenex $\wp$ that allows $\forall$E to be applied to replace all x variables with appropriate y variables according to σ. Because σ is generated from `sub`, this is sufficient to unify all literals from `L`. Consequently, any disjunct of the DNF matrix of D_i^{***} contains an explicit contradiction. Therefore, the unambiguity of σ is a sufficient criterion for a $\wedge$I-minimal proof once $\mathtt{P}_\wp$ is considered.

[13]The algorithm for generating substitution lists σ concerns the substitution of x variables from a *minimal set* of unifiable pairs of literals in step 2 of the FOL-Decider. This algorithm must be distinguished from Algorithm 9.1 described in section 9 of [Lampert (2019)], which generates *maximal* substitution lists for a *single* pair of connected literals. The generation of substitution lists σ in step 2 of the FOL-Decider serves not for deciding whether connected literals are unifiable but rather for deciding whether further $\wedge$I applications are necessary in the search for a $\wedge$I-minimal proof along a given proof path.

Thus, $\wedge$I must be applied only if σ is not unambiguous. If $\mathtt{P}_\wp$ is not satisfied for an x variable ν, then ν must additionally be replaced with y_0 in order to satisfy $\mathtt{P}_\wp$. By definition, $\exists y_0$ precedes any optimized prenex $\wp$. A specification `sub` of substitutions considers $\mathtt{P}_\wp$ as soon as σ is unambiguous apart from satisfying $\mathtt{P}_\wp$.[14] The FOL-Decider considers all combinatorial possibilities for the introduction of y_0 to satisfy $\mathtt{P}_\wp$ (for an illustration, cf. Example 5.28).[15] Thus, introducing y_0 for the sake of satisfying $\mathtt{P}_\wp$ results in further alternative specifications `sub` and, consequently, alternative substitution lists σ, thus necessitating alternative proof paths. These alternative proof paths consider all combinatorial variants of minimal sets `L` of unifiable pairs of literals, optimized prenexes $\wp$ and alternative substitution specifications `sub`. If the substitution of x variables with y_0 is no longer necessary to satisfy $\mathtt{P}_\wp$ subsequent to $\wedge$I applications, this is also considered via corresponding modifications of `sub` (cf. *exception 2* on p. 26).

An optimized prenex $\wp$ is *optimal* iff σ is unambiguous given that $\mathtt{P}_\wp$ is considered.

Definition 5.22. An optimized prenex $\wp$ is *optimal* iff each x variable μ in $\wp$ is to the right of the y variable ν that is to replace it according to the unambiguous substitution list σ.

5.3. **$\wedge$I Applications.** Applications of $\wedge$I are necessary only if σ is ambiguous. In this case, at least one x variable μ in `sub` must be multiplied to enable the substitution of $\mu_1, \ldots, \mu_n$ with n different y variables.

Definition 5.23. An x variable μ *is multiplied* iff the expression $\forall\mu A(\mu)$ in D_i^* is multiplied by applying $\wedge$I.

To avoid the application of DIS2 when generating anti-prenex normal forms in step 2 of the FOL-Decider, it is, strictly speaking, not only the expression $\forall\mu A(\mu)$ that is multiplied; instead, what is multiplied is the entire expression that contains all universal quantifiers and occurrences of $\vee$ preceding $\forall\mu A(\mu)$ in the logical hierarchy of D_i^* up to the next occurrence of some existential quantifier or $\wedge$. This multiplied expression will always be preceded by a universal quantifier (cf., e.g., Example 5.10 in footnote 11: to multiply x_6, the expression preceded by $\forall x_4$ is multiplied, as seen in the last line in (5.13)). For simplicity, I will still use $\forall\mu A(\mu)$ to refer to the expression that is multiplied, which may be preceded by universal quantifiers other than $\forall\mu$.

$\forall\mu A(\mu)$ is multiplied $n-1$ times by applying $\wedge$I to generate n conjuncts such that μ_i is replaced with the i-th of the n y variables with which μ must be replaced according to σ. To distinguish among different $\wedge$I applications for the multiplication of different x variables, I refer to all of the $n-1$ applications of $\wedge$I that are necessary to multiply *one* x variable μ as "*one* $\wedge$I application". The steps along a proof path are numbered with respect to the $\wedge$I applications that are performed to multiply x variables. Although x variables ν ($\nu \neq \mu$) that are bound by universal quantifiers occurring in $\forall\mu A(\mu)$ are also multiplied when multiplying $\forall\mu A(\mu)$, only μ is "*the* multiplied x variable", because the purpose of applying $\wedge$I is to replace instances of μ in different conjuncts with different y variables.

[14] If $\mathtt{P}_\wp$ were to be considered in previous proof steps with ambiguous substitution lists σ, the necessity to introduce substitutions of x variables with y_0 would not be kept to a minimum. Example 5.10 in footnote 11 on p. 16 illustrates this. Given an optimized prenex in which x_3 is to the left of y_1 and y_2, x_3 would have to be additionally replaced with y_0. However, this additional substitution is not necessary for the $\wedge$I-minimal proof described in Example 5.10.

[15] Again, it might be possible to define criteria to restrict the possible combinations without excluding any $\wedge$I-minimal proofs. However, no such criteria are defined in the FOL-Decider for now.

Definition 5.24. *One* $\wedge$I application consists of the replacement of the universally quantified expression $\forall\mu A(\mu)$ with a conjunction of n conjuncts $\forall\mu_1 A(\mu_1), \ldots, \forall\mu_n A(\mu_n)$, in which all variables bound by quantifiers of the i-th conjunct are subscripted with i for rectification.

In the course of the $\wedge$I application, $\mathtt{L}, \wp$, and $\mathtt{sub}$ must be modified and extended. In the following, I distinguish $\mathtt{L}$ and $\mathtt{sub}$ prior to a $\wedge$I application from their *modifications* $\mathtt{L}'$ and $\mathtt{sub}'$ directly subsequent to the $\wedge$I application and their *extensions* $\mathtt{L}''$ and $\mathtt{sub}''$ prior to possible further $\wedge$I applications. The resulting $\mathtt{L}''$ and $\mathtt{sub}''$ subsequent to a given $\wedge$I application are identical to the $\mathtt{L}$ and $\mathtt{sub}$ prior to the next $\wedge$I application. In the case of $\wp$, I distinguish $\wp$ prior to a $\wedge$I application from the subsequent *modified and extended* $\wp'$ prior to possible further $\wedge$I applications. However, I will speak of $\mathtt{L}$, $\wp$, and $\mathtt{sub}$ in general if I am not considering their modifications or extensions.

Subsequent to a $\wedge$I application, the FOL-Decider first modifies $\mathtt{L}$ (resulting in $\mathtt{L}'$) and $\mathtt{sub}$ (resulting in $\mathtt{sub}'$). However, if $\forall\mu A(\mu)$ contains $\vee$, then $\mathtt{L}'$ must also be extended subsequent to the $\wedge$I application (cf. section 5.4 for details). This extension results in $\mathtt{L}''$ and is considered only after the modification of $\mathtt{L}$ and $\mathtt{sub}$ to $\mathtt{L}'$ and $\mathtt{sub}'$. During the generation of $\mathtt{sub}$, $\mathtt{sub}'$ and $\mathtt{sub}''$, the substitutions of x variables in positions other than xx positions are specified prior to those in xx positions, as the latter substitutions depend on the former. However, I omit these details in the following. $\wp$ is modified and extended (resulting in $\wp'$) only after $\mathtt{L}''$ is generated because it refers to $\mathtt{L}''$. Finally, $\mathtt{sub}'$ is extended to $\mathtt{sub}''$; this extension is performed last because it depends on $\mathtt{L}''$ and $\wp'$. These modifications and extensions are explained in this and the following section.

Subsequent to the application of $\wedge$I, the index depths of variables bound by quantifiers occurring in $\forall\mu A(\mu)$ are increased by 1. The modification of $\mathtt{L}$, $\mathtt{sub}$ and $\wp$ basically concerns the replacement of "ancestors" with "descendants".

Definition 5.25. Let v be a variable bound by a quantifier in $\forall\mu A(\mu)$, and let v_1 to v_n denote the variables resulting from the $\wedge$I application. Then, the variables $v_1, \ldots, v_n$ are *descendants of* v, and v is their *ancestor*.

$\mathtt{L}'$ and $\mathtt{sub}'$ consider the replacement of ancestors with descendants.

The FOL-Decider orders the y variables with which μ must be replaced according to σ in a list $\mathtt{yvars}$ that begins with y_0. μ_i in the i-th conjunct must then be replaced with the i-th y variable from $\mathtt{yvars}$. Thus, the corresponding literals in $\mathtt{L}$, in which μ must be replaced with the i-th y variable from $\mathtt{yvars}$, are replaced with the literals from the i-th conjunct in D_i^*. Consequently, all variables of these literals that are bound by quantifiers occurring in $\forall\mu_i A(\mu_i)$ are subscripted with i in $\mathtt{L}'$. The substitutions in $\mathtt{sub}$ are adjusted likewise to obtain the modified $\mathtt{sub}'$.

In addition to literals containing μ, literals in $\mathtt{L}$ that do not contain μ but do contain variables bound by quantifiers occurring in $\forall\mu A(\mu)$ must be modified subsequent to the $\wedge$I application. All combinatorial possibilities for replacing literals in $\mathtt{L}$ that do not contain μ but do contain variables bound by quantifiers occurring in $\forall\mu A(\mu)$ with literals from different conjuncts of the result of the $\wedge$I application are realized in alternative sets $\mathtt{L}'$ and considered in the respective specifications $\mathtt{sub}'$. The only requirements are that the pairs of literals still be unifiable and that the selection of literals does not violate $\mathtt{FP}$. Alternative selections of literals from the n conjuncts give rise to alternative proof paths.

Furthermore, all alternative ways of subscripting a y variable ν that replaces x variables in xx positions according to $\mathtt{sub}$ and that is bound by an existential quantifier occurring in $\forall\mu A(\mu)$ are realized in alternative substitution specifications $\mathtt{sub}'$. Such alternatives arise

in the case that the corresponding x variables must be replaced with different descendants ν_j and ν_k ($j, k \in \mathbb{N}$ and $\leq n$, $j \neq k$) of the ancestor y variable ν in $\mathtt{sub}'$ in positions that are not xx positions. It might be possible to define criteria to restrict the number of alternative sets $\mathtt{L}'$ and alternative specifications $\mathtt{sub}'$ without excluding $\wedge$I-minimal proofs. However, no such criteria are defined in the FOL-Decider for now. This is another reason why an enormous number of alternative proof paths are generated in step 2 of the FOL-Decider.

The differences between $\mathtt{sub}$ and $\mathtt{sub}'$ described so far concern the replacement of ancestors with descendants. However, there is one additional difference (= *exception 1*). Let ν ($\nu \neq \mu$) be an x variable occurring in $\forall\mu A(\mu)$. Let ν be replaced with some y variable ρ in some position that is not an xx position (*case 1* on p. 14), and consequently, let ν be replaced with ρ in some other position that is an xx position (*case 2.1* on p. 14). Since ν is multiplied in the course of the $\wedge$I application, it may well be that its descendant ν_i no longer needs to be replaced with ρ (or some descendant ρ_j) in some position that is not an xx position due to $\mathtt{L}'$. Consequently, there is also no need to replace ν_i with ρ (or some descendant ρ_j) in xx positions, as long as this is also not required due to other x variables that share some xx position with ν_i. This possibility is considered when modifying $\mathtt{sub}$, to the effect that the requirement to replace ν_i in xx positions with ρ (or some descendant ρ_j) is discarded in $\mathtt{sub}'$. This keeps the number of required substitutions and, consequently, the number of necessary $\wedge$I applications low.

In addition to $\mathtt{L}$ and $\mathtt{sub}$, $\wp$ must also be modified and extended subsequent to $\wedge$I applications. First of all, this is required because quantifiers from $\forall\mu A(\mu)$ are multiplied. Furthermore, additional quantifiers may need to be considered in the course of extending $\mathtt{L}'$ to $\mathtt{L}''$ (cf. section 5.4 for details). In the following, however, I abandon the term "modification of $\wp$" and simply speak of "extensions of $\wp$", including the replacement of ancestors with their descendants. Before explaining extensions of $\mathtt{L}'$, I specify the *Principle of Prenex Extension* $\mathtt{PE}\wp$ because the principle that guides the extension of $\wp$ depends simply on the generation of all optimized prenexes $\wp+$ involving the variables in $\mathtt{L}''$ (however $\mathtt{L}''$ is generated). From all these prenexes $\wp+$, the FOL-Decider selects only those prenexes $\wp'$ that satisfy the following principle:

Principle of Prenex Extension ($\mathtt{PE}\wp$). *Prenex extensions $\wp'$ must maintain the possibilities provided by $\wp$ for replacing x variables with y variables by means of $\forall E$.*

The requirement of *maintaining* substitution possibilities is specified by the following three conditions:

(1) A y variable ν must not appear to the right of an x variable μ in $\wp'$ if ν is to the left of μ in $\wp$ unless $\exists\nu$ is in the scope of $\forall\mu$ in D_i^* after the extension of $\mathtt{L}'$ to $\mathtt{L}''$.[16]

[16]This exception is necessary because optimized prenexes are generated in relation to *purged** D_i^*. The following example illustrates this:

$$\forall x_1\exists y_1(\exists y_2(\exists y_3(Gy_1y_3 \wedge Gy_2y_3) \wedge \exists y_4\forall x_3\neg Hy_4y_2x_3) \wedge \forall x_4Hy_1x_4x_1) \wedge$$
$$\forall x_2(\forall x_5\forall x_8Hx_5x_2x_8 \vee \forall x_6\forall x_9\neg Hx_2x_6x_9 \vee \forall x_7\neg Gx_2x_7) \tag{5.33}$$

One of the two minimal sets $\mathtt{L}$ of (5.33) is $\{\{Gy_2y_3,\ \neg Gx_2x_7\},\ \{Hx_5x_2x_8,\ \neg Hy_4y_2x_3\},\ \{Hy_1x_4x_1,\ \neg Hx_2x_6x_9\}\}$. With regard to this $\mathtt{L}$, Gy_1y_3 is deleted during the process of purging. This results in the following optimized prenex $\wp$:

$$\{y_2, y_3, y_4, x_1, y_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\} \tag{5.34}$$

However, the second conjunct of (5.33) must be multiplied to prove the refutability of (5.33) because x_2 must be replaced with both y_1 and y_2 in the considered $\mathtt{L}$. In the course of the $\wedge$I application, $\mathtt{L}'$ must be

(2) For each ancestor v occurring in $\wp$, generate the set of its descendants that occur in $\wp+$. Let k be the number of resulting non-empty sets of descendants, and let r_1 to r_k be their cardinalities. For these sets, generate their Cartesian product. Each member of the Cartesian product contains exactly one descendant for each ancestor. For each of the $r_1 \cdot \ldots \cdot r_k$ members m_i of the Cartesian product, generate the sequence $\wp^*$ from $\wp+$ by replacing the descendants of m_i with their ancestor. $\wp+$ is a prenex $\wp'$ that satisfies $\mathtt{PE}\wp$ only if at least one of the resulting $r_1 \cdot \ldots \cdot r_k$ prenexes $\wp^*$ satisfies condition (1) compared to $\wp$. This test ensures that for each ancestor in $\wp$, at least one descendant takes over its relative position in $\wp'$.

(3) Each descendant μ_i of the multiplied x variable μ must be to the right of the y variable with which μ_i must be replaced according to $\mathtt{sub}'$, if $\mathtt{P}_\wp$ is satisfied prior to the $\wedge$I multiplication.[17]

$\mathtt{PE}\wp$ ensures that the necessity to replace x variables additionally with y_0 is kept rare by the extension of $\wp$. Furthermore, condition (3) of $\mathtt{PE}\wp$ ensures that all replacements of μ_1 to μ_n with the respective y variables from $\mathtt{sub}'$ can be achieved by applying $\forall$E to a

extended by $\{Gy_1y_3, \neg Gx_{2_2}x_{7_2}\}$. From this, it follows that Gy_1y_3 is now part of D_i^*. Consequently, y_2, y_3, and y_4 cannot be to the left of x_1 in $\wp'$.

[17]The if-clause "if $\mathtt{P}_\wp$ ..." is necessary because in the search for proofs, x variables are multiplied without considering the *Prenex Condition* $\mathtt{P}_\wp$ as long as σ is ambiguous (cf. p. 19 and footnote 14). This is illustrated in the following example.

Example 5.26. Let D_i be given as follows:

$$\forall x_1 \exists y_1 (\forall x_2 \forall x_6 F x_2 x_1 y_1 x_6 \wedge \forall x_3 \forall x_7 G x_3 x_1 y_1 x_7 \wedge \forall x_4 \forall x_8 H x_4 x_1 y_1 x_8 \wedge \forall x_5 (\exists y_2 \forall x_9 \neg F x_5 y_2 x_9 y_1 \vee \exists y_3 \forall x_{10} \neg G x_5 y_3 x_{10} y_1 \vee \forall x_{11} \forall x_{12} \neg H x_5 x_{11} x_{12} y_1)) \tag{5.35}$$

The FOL-Decider finds a $\wedge$I-minimal proof in which the final D_i^* is as follows:

$$\begin{gathered} \forall x_{1_1} \exists y_{1_1} \forall x_{2_1} \forall x_{6_1} F x_{2_1} x_{1_1} y_{1_1} x_{6_1} \wedge \\ \forall x_{1_{2_1}} \exists y_{1_{2_1}} (\forall x_{4_{2_1}} \forall x_{8_{2_1}} H x_{4_{2_1}} x_{1_{2_1}} y_{1_{2_1}} x_{8_{2_1}} \wedge \\ \forall x_{5_{2_1}} (\exists y_{2_{2_1}} \forall x_{9_{2_1}} \neg F x_{5_{2_1}} y_{2_{2_1}} x_{9_{2_1}} y_{1_{2_1}} \vee \\ \exists y_{3_{2_1}} \forall x_{10_{2_1}} \neg G x_{5_{2_1}} y_{3_{2_1}} x_{10_{2_1}} y_{1_{2_1}} \vee \\ \forall x_{11_{2_1}} \forall x_{12_{2_1}} \neg H x_{5_{2_1}} x_{11_{2_1}} x_{12_{2_1}} y_{1_{2_1}})) \wedge \\ \forall x_{1_{2_2}} \exists y_{1_{2_2}} \forall x_{3_{2_2}} \forall x_{7_{2_2}} G x_{3_{2_2}} x_{1_{2_2}} y_{1_{2_2}} x_{7_{2_2}} \end{gathered} \tag{5.36}$$

(5.36) results from a first application of $\wedge$I to replace descendants of x_1 with descendants of y_2 and y_3 and a second application of $\wedge$I to additionally replace a descendant of x_{1_2} with y_0 in order to satisfy $\mathtt{P}_\wp$. This $\wedge$I application would be omitted if one were to neglect the if-clause in condition (3). The $\wedge$I-minimal proof of contradiction is based on the following final $\mathtt{L}$, σ, and $\wp$.

$\mathtt{L}$:

$$\{\{F x_{2_1} x_{1_1} y_{1_1} x_{6_1}, \neg F x_{5_{2_1}} y_{2_{2_1}} x_{9_{2_1}} y_{1_{2_1}}\}, \tag{5.37}$$

$$\{H x_{4_{2_1}} x_{1_{2_1}} y_{1_{2_1}} x_{8_{2_1}}, \neg H x_{5_{2_1}} x_{11_{2_1}} x_{12_{2_1}} y_{1_{2_1}}\}, \tag{5.38}$$

$$\{G x_{3_{2_2}} x_{1_{2_2}} y_{1_{2_2}} x_{7_{2_2}}, \neg G x_{5_{2_1}} y_{3_{2_1}} x_{10_{2_1}} y_{1_{2_1}}\}\} \tag{5.39}$$

σ:

$$\{\{x_{1_1}, y_{2_{2_1}}\}, \{x_{2_1}, y_0\}, \{x_{6_1}, y_{1_{2_1}}\}, \{x_{1_{2_1}}, y_0\}, \{x_{1_{2_2}}, y_{3_{2_1}}\}, \{x_{3_{2_2}}, y_0\}, \{x_{4_{2_1}}, y_0\}, \{x_{5_{2_1}}, y_0\}, \{x_{7_{2_2}}, y_{1_{2_1}}\}, \{x_{8_{2_1}}, y_{1_{2_1}}\}, \{x_{9_{2_1}}, y_{1_1}\}, \{x_{10_{2_1}}, y_{1_{2_2}}\}, \{x_{11_{2_1}}, y_0\}, \{x_{12_{2_1}}, y_{1_{2_1}}\}\} \tag{5.40}$$

$\wp$:

$$\{y_0, x_{1_{2_1}}, y_{1_{2_1}}, x_{5_{2_1}}, y_{2_{2_1}}, y_{3_{2_1}}, x_{1_1}, y_{1_1}, x_{1_{2_2}}, y_{1_{2_2}}, x_{2_1}, x_{6_1}, x_{8_{2_1}}, x_{4_{2_1}}, x_{9_{2_1}}, x_{10_{2_1}}, x_{12_{2_1}}, x_{11_{2_1}}, x_{3_{2_2}}, x_{7_{2_2}}\} \tag{5.41}$$

corresponding prenex normal form D_i^{**} given that $\mathtt{P}_\wp$ is satisfied prior to the $\wedge$I application. This was the stated purpose of the $\wedge$I application.

Remark 5.27. $\mathtt{PE}\wp$ is not a necessary principle for a $\wedge$I-minimal proof search. For a decision procedure for D_i based on a $\wedge$I-minimal proof search, it is sufficient to assume the finiteness of the combinatorial possibilities for generating prenexes (cf. section 8).

Alternative possibilities for extending $\wp$ to $\wp'$ are considered on alternative proof paths. The modifications and extensions of $\mathtt{L}$, $\wp$, and $\mathtt{sub}$ consider all alternatives that might result in a $\wedge$I-minimal proof.

Example 5.28. This example illustrates how a $\wedge$I-minimal proof is found when considering substitutions with y_0 to satisfy condition $\mathtt{P}_\wp$. Extensions of $\mathtt{L}'$ (cf. section 5.4) need not be considered in this example.

Let D_i be given as follows:

$$\forall x_1 \exists y_1 F x_1 y_1 \wedge \forall x_2 (\forall x_3 \neg F x_2 x_3 \vee \forall x_4 \neg F x_4 x_2) \tag{5.42}$$

There is exactly one minimal set $\mathtt{L}$ of unifiable pairs of literals in the case of (5.42):

$$\{\{F x_1 y_1, \neg F x_2 x_3\}, \tag{5.43}$$
$$\{F x_1 y_1, \neg F x_4 x_2\}\} \tag{5.44}$$

Also, there is exactly one optimized prenex $\wp$:

$$\{x_1, y_1, x_2, x_3\} \tag{5.45}$$

x_3 must be replaced with y_1 in (5.43), and x_2 must be replaced with y_1 in (5.44). x_1 and x_2 must be replaced with the same y variable in (5.43), and x_1 and x_4 must be replaced with the same y variable in (5.44). If $\mathtt{P}_\wp$ is not considered, then x_1 and, consequently, x_4 must be replaced only with y_1 because x_2 must be replaced with y_1 in (5.44). Thus, σ is unambiguous as long as $\mathtt{P}_\wp$ is not considered. However, $\mathtt{P}_\wp$ is not satisfied because y_1 is not to the left of x_1 in (5.45). $\forall$E cannot be applied to replace x_1 with y_1 in the corresponding prenex normal form with the optimized prenex given in (5.45). Therefore, x_1 must additionally be replaced with y_0 in order to satisfy $\mathtt{P}_\wp$. Either (1) x_1 and, consequently, x_2 must be replaced with y_0 in (5.43); (2) x_1 and, consequently, x_4 must be replaced with y_0 in (5.44); or (3) both. For completeness, I also note the special case (4), which specifies that x_1 must be replaced with y_0 in addition to y_1 but not in the xx positions in (5.43) and (5.44). This case is also considered in the FOL-Decider. Cases (1) to (4) give rise to the following 4 substitution lists, $\sigma(1)$ to $\sigma(4)$:

(1) $\sigma(1) = \{\{x_1, y_0\}, \{x_2, y_0, y_1\}, \{x_3, y_1\}, \{x_4, y_0\}\}$,
(2) $\sigma(2) = \{\{x_1, y_0, y_1\}, \{x_2, y_1\}, \{x_3, y_1\}, \{x_4, y_0\}\}$,
(3) $\sigma(3) = \{\{x_1, y_0, y_1\}, \{x_2, y_0, y_1\}, \{x_3, y_1\}, \{x_4, y_1\}\}$,
(4) $\sigma(4) = \{\{x_1, y_0, y_1\}, \{x_2, y_1\}, \{x_3, y_1\}, \{x_4, y_1\}\}$.

These 4 variants correspond to 4 alternative proof paths. The special case (4), with $\sigma(4)$, does not result in a $\wedge$I-minimal proof because x_{1_1} is not replaced with y_0 in $F x_{1_1} y_{1_1}$. Instead, x_{1_2} is replaced with y_{1_1} in $F x_{1_2} y_{1_2}$. Consequently, $\mathtt{L}'$ does not contain any literal from the conjunct $\forall x_{1_1} \exists y_{1_1} F x_{1_1} y_{1_1}$ that results from multiplying $\forall x_1 \exists y_1 F x_1 y_1$. This contradicts the *Fundamental Principle* $\mathtt{FP}$ (cf. p. 13). Therefore, this proof path terminates because it contradicts a principle of the $\wedge$I-minimal proof search. The FOL-Decider first realizes all

combinatorial possibilities (1) to (4) and, consequently, $\sigma(1)$ to $\sigma(4)$. The proof path based on $\sigma(4)$ is then abandoned subsequent to the $\wedge$I application because it violates FP.

In the cases of variants (1) and (3), $\sigma(1)$ and $\sigma(3)$ cause it to be necessary to multiply x_2. This does not result in a $\wedge$I-minimal proof. Therefore, let us consider only variant (2), with $\sigma(2)$. According to $\sigma(2)$, only x_1 must be multiplied. This results in the following D_i^*:

$$\forall x_{1_1}\exists y_{1_1}Fx_{1_1}y_{1_1} \wedge \forall x_{1_2}\exists y_{1_2}Fx_{1_2}y_{1_2} \wedge \forall x_2(\forall x_3\neg Fx_2x_3 \vee \forall x_4\neg Fx_4x_2) \tag{5.46}$$

A $\wedge$I-minimal proof is found if one replaces x_{1_1} with y_0 to unify $\{Fx_{1_1}y_{1_1}, \neg Fx_4x_2\}$ and replaces x_{1_2} with y_{1_1} to unify $\{Fx_{1_2}y_{1_2}, \neg Fx_2x_3\}$. This proof rests on the unification of the following pairs of literals from (5.46):

$$\{\{Fx_{1_1}y_{1_1}, \neg Fx_4x_2\}, \{Fx_{1_2}y_{1_2}, \neg Fx_2x_3\}\} \tag{5.47}$$

It also rests on the following substitution list σ:

$$\{\{x_{1_1}, y_0\}, \{x_{1_2}, y_{1_1}\}, \{x_2, y_{1_1}\}, \{x_3, y_{1_2}\}, \{x_4, y_0\}\} \tag{5.48}$$

The corresponding optimal prenex $\wp$ is as follows:

$$\{y_0, x_{1_1}, y_{1_1}, x_{1_2}, y_{1_2}, x_2, x_3, x_4\} \tag{5.49}$$

If one generates the corresponding optimal prenex form D_i^{**} from (5.46) with the optimal prenex given in (5.49), then one can deduce an explicit contradiction D_i^{***} by applying $\forall$E.

5.4. **Extending $\mathtt{L}'$ and $\mathtt{sub}'$.** If the multiplied expression $\forall\mu A(\mu)$ contains $\vee$, then $\mathtt{L}'$, $\wp$, and $\mathtt{sub}'$ must be extended.

According to conditions (i) and (ii) in Definition 5.4, the following two principles hold for the extension of $\mathtt{L}'$.

$\perp$-Principle ($\perp$P). $\mathtt{L}''$ *must satisfy the condition that each disjunct of the DNF matrix of* D_i^* *contains two conjuncts,* A *and* $\neg A$*, given that all pairs of literals in* $\mathtt{L}''$ *are unified.*

Minimal L-Principle ($\mathtt{M_L}$P). $\mathtt{L}''$ *is* minimally sufficient[18] *to satisfy* $\perp$P.

Both principles also hold for any L prior to any application of $\wedge$I. $\mathtt{M_L}$P avoids unnecessary $\wedge$I applications that may violate the *Fundamental Principle* FP of the $\wedge$I-minimal proof search strategy.

In addition to applying the *Fundamental Principle* FP, the FOL-Decider restricts the selection of literals in L in accordance with the following principle for an effective $\wedge$I-minimal proof search:

[18]This term is defined as follows:

Definition 5.29. A set S is minimally sufficient to satisfy a certain condition C iff no proper subset of S is sufficient to satisfy C.

Principle of L-Extension (PEL). *All literals from* $\mathtt{L}'$ *must be maintained when extending* $\mathtt{L}'$ *to* $\mathtt{L}''$*. Through all modifications and extensions, literals from* L*, once they are members of* L*, are modified only by increasing the index depths of their variables.*

This principle ensures that any selection of a literal in the course of the proof search must be necessary for the ∧I-minimal proof. It follows that along a given proof path, the size of L and the number of selected conjuncts can only increase.

The FOL-Decider generates the initial L and any extensions $\mathtt{L}''$ in accordance with the principles FP, $\bot$P, $\mathtt{M_L}$P, and PEL.

Similar to the principles PE$\wp$ and PEL, the substitutions in $\mathtt{sub}'$ must also be maintained in $\mathtt{sub}''$. An exception can be found in certain substitutions of x variables μ with y_0. Let μ be an x variable that must be replaced with y_0 according to $\mathtt{sub}'$ in order to satisfy the *Prenex Condition* $\mathtt{P}_\wp$ with respect to $\wp$. Let $\mathtt{L}'$ be extended such that μ must additionally be replaced with a y variable ν that is to the left of μ according to $\wp'$. In this case, ν can take over the role of y_0, meaning that μ no longer needs to be replaced with y_0. This is *exception 2* of the maintenance principle for the specification of substitutions sub. The FOL-Decider considers this exception and computes its consequences for the substitutions of x variables in xx positions. Like *exception 1* (cf. p. 22), *exception 2* minimizes the length of the partial list in σ, and therefore the need to multiply x variables, without excluding any ∧I-minimal proofs.

Another, final, exception corresponds to substitutions at xx positions in pairs of literals in $\mathtt{L}'$. If x variables in these positions must be replaced with further y variables due to the extension of $\mathtt{L}'$ to $\mathtt{L}''$, then *additional* alternative specifications $\mathtt{sub}''$ must be generated that consider alternative substitutions of x variables in xx positions in pairs of literals from $\mathtt{L}'$ (*exception 3*). Thus, not all replacements of x variables with y variables in xx positions are maintained in these *additional* specifications $\mathtt{sub}''$. *All* alternative substitution possibilities are considered on alternative proof paths to ensure that a ∧I-minimal proof path is found if one exists.

Principle of sub-Extension (PEsub). *Apart from* exception 1 *as described on p. 22 and* exceptions 2 *and* 3 *as introduced above, all substitutions must be maintained subsequent to* ∧*I applications: through all modifications and extensions, variables from* sub *are modified only by increasing their index depths, apart from the mentioned exceptions.*

The mentioned *exceptions* are justified by the intent to find a ∧I-minimal proof, if it exists, without generating superfluous proof paths.

I summarize the maintenance principles for extending $\mathtt{L}'$, $\wp$, and $\mathtt{sub}/\mathtt{sub}'$ into the following single principle:

Principle of Extension (PE). L, $\mathtt{L}'$, $\wp$, sub *and* $\mathtt{sub}'$ *must be modified and extended in accordance with* PEL, PE$\wp$ *and* PEsub.

Alternative minimal sets of pairs of literals, alternative specifications of substitutions and alternative optimized prenexes are considered on alternative proof paths. The FOL-Decider considers only extensions of L, sub, and $\wp$ that are in accordance with PE. Any selection of pairs of literals and any specification of substitutions with respect to some optimized prenex is, roughly speaking, a fixed constituent in the search for a ∧I-minimal proof along a proof path. On the one hand, this limits the search for proofs in the NNF-calculus to an effective search for ∧I-minimal proofs. On the other hand, all combinatorial

possibilities consisting of different alternatives for L, $\wp$ and sub are generated to ensure that a $\wedge$I-minimal proof of contradiction for D_i is found if D_i is refutable.

Remark 5.30. It is not necessary to reduce the search for a $\wedge$I-minimal proof to a search along proof paths on which $\mathtt{M_L P}$ and PE are applied. For a decision procedure based on the $\wedge$I-minimal proof strategy, it is sufficient to rely on a *finite* search space that *contains* the proof paths that are generated in accordance with $\mathtt{M_L P}$ and PE. In this sense, these principles are not necessary but rather are principles for an *effective* $\wedge$I-minimal proof search (cf. section 8).

The principles mentioned thus far may already constitute a sat-proof for a D_i. The following is a simplest possible example of such a proof.

Example 5.31. Let D_i be given as follows:

$$\exists y_1 F y_1 \wedge \exists y_2 G y_2 \wedge \forall x_1(\neg F x_1 \vee \neg G x_1) \tag{5.50}$$

There are only one optimized prenex $\wp$ and only one sub/L:

$$\wp: \quad \{y_1, y_2, x_1\} \tag{5.51}$$

$$\begin{aligned} \mathtt{sub/L}: \quad & \{\{\{\{x_1, y_1\}\}, \{\}, \{\}\}, \{F y_1, \neg F x_1\}\}, \\ & \{\{\{\{x_1, y_2\}\}, \{\}, \{\}\}, \{G y_2, \neg G x_1\}\}\} \end{aligned} \tag{5.52}$$

From the substitution specification sub given in (5.52), the following substitution list σ is obtained:

$$\sigma: \quad \{\{x_1, y_1, y_2\}\} \tag{5.53}$$

Thus, $\forall x_1(\neg F x_1 \vee \neg G x_1)$ in (5.50) must be multiplied to make it possible to replace x_{1_1} with y_1 and x_{1_2} with y_2:

$$\exists y_1 F y_1 \wedge \exists y_2 G y_2 \wedge \forall x_{1_1}(\neg F x_{1_1} \vee \neg G x_{1_1}) \wedge \forall x_{1_2}(\neg F x_{1_2} \vee \neg G x_{1_2}) \tag{5.54}$$

This results in the following modification of (5.52):

$$\begin{aligned} \mathtt{sub'/L'}: \quad & \{\{\{\{x_{1_1}, y_1\}\}, \{\}, \{\}\}, \{F y_1, \neg F x_{1_1}\}\}, \\ & \{\{\{\{x_{1_2}, y_2\}\}, \{\}, \{\}\}, \{G y_2, \neg G x_{1_2}\}\}\} \end{aligned} \tag{5.55}$$

According to PEL, all pairs of literals from (5.55) must be maintained when extending (5.55). To satisfy $\perp$P, (5.55) must be extended by either $\{G y_2, \neg G x_{1_1}\}$ or $\{F y_1, \neg F x_{1_2}\}$. However, both extensions violate $\mathtt{M_L P}$. If one were to select $\{G y_2, \neg G x_{1_1}\}$ in addition to $\{F y_1, \neg F x_{1_1}\}$, then $\{G y_2, \neg G x_{1_2}\}$ would be superfluous. Consequently, the $\wedge$I application for multiplying x_1 would be superfluous. This would violate the *Fundamental Principle* FP, because $\forall x_{1_2}(\neg F x_{1_2} \vee \neg G x_{1_2})$ would no longer be selected. A similar situation would result if one were to select $\{F y_1, \neg F x_{1_2}\}$ in addition to $\{G y_2, \neg G x_{1_2}\}$. Thus, the proof along the described proof path terminates because every extension violates some principle of the $\wedge$I-minimal proof strategy. Therefore, (5.50) is proven to be satisfiable (not refutable) since there are no alternatives to (5.51) and (5.52). No $\wedge$I-minimal proof path can be found. If (5.50) were refutable, however, a $\wedge$I-minimal proof would exist, according to Theorem 5.2.

6. Termination: Loop List

$\wedge$I multiplies an x variable μ. The multiplication of an x variable μ may necessitate additional multiplications of other x variables ν. A loop list codes sequences of such causally related multiplications of x variables.

To identify which multiplications of x variables ν are necessitated by the multiplication of an x variable μ, one must compare the list of substitutions prior to the last $\wedge$I multiplication ($\sigma 1$) with the substitution list after that $\wedge$I application ($\sigma 2$) on each proof path. After each application of $\wedge$I, the FOL-Decider generates new lists of the parameters sub/L and ρ. Alternatives of sub, L or ρ generate alternative proof paths. Alternative modifications may cause alternative substitution lists $\sigma 2$. For each proof path, the modification of the loop list on this path depends on the comparison between the list of substitutions prior to the last $\wedge$I multiplication ($\sigma 1$) with the substitution list on the actual path after that $\wedge$I application ($\sigma 2$). Since only variables that must be replaced with more than one y variable are considered, all partial lists of length 2 are deleted from $\sigma 1$ and $\sigma 2$. $\sigma 1$ can be generated from the sub prior to the $\wedge$I application[19], and $\sigma 2$ can be generated from the sub$''$ subsequent to the $\wedge$I application. Furthermore, it must be considered that L$''$ and sub$''$ contain the descendants of variables v bound by quantifiers from the multiplied expression $\forall\mu A(\mu)$. Let $\mathtt{vars_d}$ be the set of all descendants, and let $\mathtt{vars_a}$ be the set of all of their ancestors.

Definition 6.1. The multiplication of an x variable μ ($= \mathtt{xvar_c}$) *necessitates* the multiplication of an x variable ν ($= \mathtt{xvar_e}$) iff one of the following conditions is satisfied:

(1) ν is not a member of $\mathtt{vars_d}$, and
   (a) according to $\sigma 2$, ν is to be replaced with a y variable yvar that is not from $\mathtt{vars_d}$, and according to $\sigma 1$, ν is not to be replaced with yvar; or
   (b) according to $\sigma 2$, ν is to be replaced with a y variable yvar from $\mathtt{vars_d}$, and according to $\sigma 1$, ν is not to be replaced with the ancestor of yvar; or
   (c) according to $\sigma 2$, ν is to be replaced with two or more y variables from $\mathtt{vars_d}$ that are descendants of one and the same y variable from $\mathtt{vars_a}$.

(2) ν is a member of $\mathtt{vars_d}$, and
   (a) ν is a descendant of μ ($= \mathtt{xvar_c}$) and occurs in $\sigma 2$; or
   (b) according to $\sigma 2$, ν is to be replaced with a y variable yvar that is not from $\mathtt{vars_d}$, and according to $\sigma 1$, ν's ancestor is not to be replaced with yvar; or
   (c) according to $\sigma 2$, ν is to be replaced with a y variable yvar from $\mathtt{vars_d}$, and according to $\sigma 1$, ν's ancestor is not to be replaced with the ancestor of yvar; or
   (d) according to $\sigma 2$, ν must be replaced with two or more y variables from $\mathtt{vars_d}$ that are descendants of one and the same y variable from $\mathtt{vars_a}$.

(3) If the multiplication of an x variable μ necessitates the multiplication of an x variable ρ and the multiplication of ρ necessitates the multiplication of ν, then μ necessitates the multiplication of ν (transitivity).

In (2)(a), the multiplication of $\mathtt{xvar_c}$ necessitates the multiplication of a descendant of the multiplied variable (cf. x_1 in Example 5.50 and x_3 in Example 6.19). In all other cases, the multiplication of $\mathtt{xvar_c}$ necessitates that some other variable that is not a descendant of $\mathtt{xvar_c}$ be replaced with a further y variable subsequent to the $\wedge$I application.

[19] In fact, the FOL-Decider generates $\sigma 1$ from the first list of the heads of partial lists of the loop list prior to the $\wedge$I application.

Definition 6.2. A *loop list* is a list of partial lists, each of which consists of a head and a body.

Definition 6.3. The *head* is a list that consists of

(1) a first list consisting of
   (a) an x variable μ that must be replaced with more than one y variable according to σ[20] and
   (b) the y variables `yvars` that must replace μ according to σ and
(2) a second list `sub`/`L` containing those pairs of literals from `L` that contain literals in the scope of $\forall\mu$ in D_i^* plus the substitution specification `sub` that specifies how the x variables of those pairs of literals are to be replaced with y variables.

Definition 6.4. The `xvar`-*head* is the head of the partial list with the x variable `xvar` in the first position in the first list of the head. The `xvar` *partial list* is the partial list with the x variable `xvar` in the first position in the first list of the head.

Definition 6.5. $\texttt{xvar}_\texttt{c}$ is an x variable μ whose multiplication necessitates the multiplication of the x variable ν ($= \texttt{xvar}_\texttt{e}$) on a proof path.

Definition 6.6. The *body* of an $\texttt{xvar}_\texttt{e}$ partial list consists of all $\texttt{xvar}_\texttt{c}$-heads that necessitate the multiplication of $\texttt{xvar}_\texttt{e}$. In stage 1 on a proof path for D_i, the body is empty.

Subsequent to the application of $\wedge$I and the generation of $\texttt{L}''$, $\wp$ and $\texttt{sub}''$, the loop list is modified. To define the modification algorithm, different types of x variables must be distinguished (cf. Definitions 6.7 - 6.9).

Definition 6.7. A variable $\overline{\texttt{mxvar}}$ satisfies all of the following conditions:

(1) $\overline{\texttt{mxvar}}$ is required to be multiplied prior to the $\wedge$I application and therefore constitutes an `xvar`-head in the loop list prior to the $\wedge$I application.
(2) $\overline{\texttt{mxvar}}$ no longer needs to be multiplied subsequent to the $\wedge$I application according to $\sigma 2$.
(3) If $\overline{\texttt{mxvar}}$ is not identical to the multiplied x variable μ and $\overline{\texttt{mxvar}}$ is a member of $\texttt{vars}_\texttt{a}$, then no descendant of $\overline{\texttt{mxvar}}$ is required to be multiplied subsequent to the $\wedge$I application according to $\sigma 2$.

Definition 6.8. A variable `newmxvar` is an $\texttt{xvar}_\texttt{e}$ that must be multiplied due to the multiplication of μ ($= \texttt{xvar}_\texttt{c}$) in the last $\wedge$I application (cf. Definition 6.1).

Definition 6.9. A variable `mxvar` satisfies all of the following conditions:

(1) `mxvar` is required to be multiplied prior to the $\wedge$I application and therefore constitutes an `xvar`-head in the loop list prior to the $\wedge$I application.
(2) Either
   (a) `mxvar` is not a member of $\texttt{vars}_\texttt{a}$ and not a `newmxvar` but still is required to be multiplied subsequent to the $\wedge$I application according to $\sigma 2$, or
   (b) `mxvar` is a member of $\texttt{vars}_\texttt{a}$ and there are k ($k > 0$) descendants of `mxvar`, each of which is not a `newmxvar` but still is required to be multiplied subsequent to the $\wedge$I application according to $\sigma 2$.

Remark 6.10. It follows from Definition 6.7 and the $\wedge$I application that the multiplied x variable μ is a $\overline{\texttt{mxvar}}$. Its descendent might be a `newmxvar` but cannot be an `mxvar`.

[20]Prior to a $\wedge$I application, $\sigma = \sigma 1$; subsequent to the $\wedge$I application and the modification of the loop list, $\sigma = \sigma 2$.

The heads of the partial lists of the loop list are modified subsequent to the ∧I application as dictated by the following algorithm.

Algorithm 6.11. Apply the following rules with regard to the different types of x variables:

Rule 1: Delete all $\overline{\texttt{mxvar}}$ partial lists.

Rule 2: Replace the mxvar partial lists as follows:

(1) If mxvar is a member of $\texttt{vars}_\texttt{a}$, then replace the mxvar partial list with k partial lists. For each of the k ($k > 0$) descendants of mxvar as identified in Definition 6.9 (2)(b), modify the mxvar-head as follows:
   (a) Replace mxvar in the first list with the descendant of mxvar.
   (b) Replace the y variables yvars in the first list with the y variables $\texttt{yvars}'$ taken from $\sigma 2$.
   (c) Replace the second list sub/L with regard to $\texttt{sub}''/\texttt{L}''$.

(2) If mxvar is not a member of $\texttt{xvars}_\texttt{a}$, then modify the mxvar-head of the mxvar partial list only in accordance with **Rule 2** (1)(b) and (c).

Rule 3: Add newmxvar partial lists to the loop list:

(1) Take the μ partial list ($= \mu - PL$[21]) from the original loop list. Add the head of $\mu - PL$ to the body $B_{\mu old}$ of $\mu - PL$. The result is the body B_μ.

(2) If
   (a) newmxvar is not a member of $\texttt{vars}_\texttt{d}$ and there is no newmxvar partial list in the loop list or
   (b) newmxvar is a member of $\texttt{vars}_\texttt{d}$ and either
      (i) there is no partial list for the ancestor of newmxvar in the loop list or
      (ii) newmxvar is a descendant of the multiplied x variable μ,

   then add a newmxvar partial list $\texttt{newmxvar} - \texttt{PL}$ to the loop list. The head of $\texttt{newmxvar} - \texttt{PL}$ is a list with the following elements:
   (a) a first list with newmxvar as the first element and a list of the y variables yvars with which newmxvar must be replaced according to $\sigma 2$ as the second element;
   (b) a second list sub/L generated from $\texttt{sub}''$ and $\texttt{L}''$ with regard to the literals in the scope of the corresponding universally quantified expression in D_i^*.

   The body of $\texttt{newmxvar} - \texttt{PL}$ is B_μ.

(3) If
   (a) newmxvar is not a member of $\texttt{vars}_\texttt{d}$ and there is a newmxvar partial list $\texttt{newmxvar} - \texttt{PL}_\texttt{old}$ in the loop list or
   (b) newmxvar is a member of $\texttt{vars}_\texttt{d}$ and there is a partial list $\texttt{newmxvar} - \texttt{PL}_\texttt{old}$ for the ancestor of newmxvar in the loop list,

   then replace $\texttt{newmxvar} - \texttt{PL}_\texttt{old}$ with a partial list $\texttt{newmxvar} - \texttt{PL}$. The head of $\texttt{newmxvar} - \texttt{PL}$ is modified in accordance with **Rule 3** (2), and the body of $\texttt{newmxvar} - \texttt{PL}$ is the concatenation of the body of $\texttt{newmxvar} - \texttt{PL}_\texttt{old}$ with B_μ.

The body of a partial list codes the sequence of ∧I multiplications that necessitate the multiplication of the x variable in the head of that partial list.

Remark 6.12. Rule 1 of Algorithm 6.11 implies that $\mu - PL$ is deleted. According to **Rule 3** (2), partial lists $\mu_i - PL$ for descendants of μ are added if μ_i is a newmxvar.

[21] I.e., the partial list of the multiplied x variable μ ($= \texttt{xvar}_\texttt{c}$).

Remark 6.13. Prior to any ∧I applications, the heads contain all x variables that must be multiplied due to the initial `sub` of the initial `L`, and the bodies of the partial lists in the loop list are empty. During the course of ∧I applications, the lengths of the *bodies* will never decrease due to Algorithm 6.11. Instead, if a `newmxvar` exists, then the length of the body B_μ will increase compared with $B_{\mu old}$, and the length of the loop list may also increase. On the other hand, if there is no `newmxvar`, then the length of the *loop list* will decrease by at least 1 according to Algorithm 6.11. Since either there is a `newmxvar` or there is not, either the length of the *loop list* will decrease or the lengths of the *bodies* of the `newmxvar` − `PL`s in the loop list will increase over consecutive ∧I applications along a proof path. Therefore, as long as `newmxvar`s result from ∧I applications, the body lengths will increase.

The loop list makes it possible to specify definite termination conditions for the proof search in step 2 of the FOL-Decider.

Termination Criterion `T1` (*False Criterion*). *If the loop list = {}, then D_i is refutable.*

`T1` is the only criterion for refutation that is applied in step 2 of the FOL-Decider. If the loop list is empty, then it is no longer necessary to multiply any x variable. Therefore, an unambiguous substitution list σ, an optimal prenex $\wp$ and a minimal set `L` of unifiable pairs of literals have been found. Thus, all literals of D_i^* that are not contained in `L` can be deleted, and the resulting purged* anti-prenex normal form can be converted into a prenex normal form D_i^{**} with the optimal prenex $\wp$. Applying ∀E such that x variables are replaced with y variables in accordance with σ results in an explicit contradiction D_i^{***}. Consequently, the search for a proof of contradiction for D_i terminates with the result `False`.

During the course of ∧I applications, a head can become part of the body of a partial list. The heads in the body of a partial list were once the heads of partial lists from previous loop lists on the proof path. To define the *Loop Criterion* for terminating proof paths that do not result in a proof, let us define the concept of "isomorphic heads". Isomorphic heads can be mapped onto each other in a one-to-one manner.

Definition 6.14. Two heads are *isomorphic* iff they are identical apart from variable indices at levels > 1 and all variables that are identical up to level 1 can be mapped onto each other in a one-to-one manner such that the heads are mapped onto each other.

In a similar manner, I will also speak of *isomorphic universally quantified expressions* and *isomorphic ∧I applications*.

Whether two heads are isomorphic is decidable by (i) deciding whether the heads are identical up to level 1 according to a canonical order and, if so, (ii) replacing variables with indices at levels > 1 with variables with indices up to level 2 in a canonical way (such that different variables are replaced with different variables with indices of depth ≤ 2) and deciding whether the resulting heads are identical according to a canonical ordering. The FOL-Decider specifies an algorithm for this task. Since the details are trivial, it is sufficient to note that deciding on isomorphism reduces the maximum index level to 2.

Termination Criterion `T2` (*Loop Criterion*). *If the loop list contains a partial list PL with a body that contains a head that is isomorphic to the head of PL, then the proof search terminates on the corresponding proof path.*

Remark 6.15. As Example 5.31 illustrates, `T2` is not necessary for the termination of proof paths. `sat`-proofs of D_i may not even require the application of `T2`. `T2` is merely sufficient to terminate the search for a proof in the case that D_i is not refutable. In the absence of `T2`,

the principles of the ∧I-minimal search strategy are not sufficient to ensure the termination of the proof search in the case of an arbitrary, non-refutable D_i.

The *Loop Criterion* concerns the case in which the multiplication of an x variable necessitates an isomorphic ∧I application, namely, one in which an isomorphic x variable must be replaced with isomorphic y variables (including y_0 with regard to $\wp$) in pairs of isomorphic literals that are to be similarly unified. This criterion represents the following situation: a ∧I application is necessary to multiply a universally quantified expression $\forall\mu A(\mu)$ in order to replace the resulting descendants of μ with different y variables in different conjuncts, and this ∧I application then necessitates an isomorphic application of ∧I to an isomorphic universally quantified expression to unify similar pairs of literals in a similar manner.

I define the *correctness of the Loop Criterion* `T2` by means of the following theorem.

Theorem 6.16. *The* Loop Criterion `T2` *causes the termination only of proof paths that are not proof paths for ∧I-minimal proofs.*

Proof. Every proof step on the proof path for a ∧I-minimal proof is necessary (cf. Definition 5.1). A proof step of a ∧I-minimal proof consists of one ∧I application that is performed to multiply an x variable. The necessity of such a multiplication is computed with regard to `sub`, which specifies how to unify the pairs of literals in `L` with regard to $\wp$ (including the introduction of y_0 variables into `sub`). ∧I applications might necessitate further ∧I applications due to the resulting modifications and extensions of `L`, $\wp$ and, consequently, `sub`. A sequence of ∧I applications that satisfies the *Loop Criterion* cannot be part of a ∧I-minimal proof path because in this case, the result of this sequence of ∧I applications is the condition for an isomorphic ∧I application. The isomorphic multiplication of an isomorphic x variable for the sake of isomorphic substitutions of isomorphic pairs of literals results in an isomorphic ∧I application. Two isomorphic ∧I applications cannot both be a necessary part of a ∧I-minimal proof because the conditions for their application are the same in both cases. Thus, the proof search either enters a loop or proceeds with some alternative ∧I applications that will avoid the loop in later proof steps. In the first case, the ∧I applications will never terminate in a proof; in the second case, the sequence of ∧I applications necessitating the isomorphic ∧I application can be omitted if a proof is found. In either case, the sequence of ∧I applications necessitating the isomorphic ∧I application is not part of a ∧I-minimal proof. Therefore, if a ∧I-minimal proof for D_i exists, it cannot depend on a sequence of ∧I applications that satisfies the *Loop Criterion*. A ∧I-minimal proof can only be found on an alternative proof path. Thus, the search for a ∧I-minimal proof on a proof path that satisfies the *Loop Criterion* can be terminated. □

Note that the *Prenex Condition* $\mathtt{P}_\wp$ is considered by introducing y_0 into `sub`. Therefore, isomorphic ∧I applications also imply the similarity of $\wp$ with regard to the conditions for these isomorphic ∧I applications.

Isomorphic ∧I applications also imply similarity with regard to the number of different variables that are derivates of the same variable and, consequently, are taken from different conjuncts. The structural conditions for these isomorphic ∧I applications are the same, and thus, one can conclude that causally related ∧I applications cannot be part of a ∧I-minimal proof.

`T1` and `T2` are the essential termination criteria.

Before we prove that these criteria suffice for terminating the ∧I proof strategy, we need to consider the generation of multiplied expressions headed by universal quantifiers

$\forall\mu_1$ and $\forall\mu_2$ that both originate from multiplying $\forall\mu$ but with non-isomorphic scopes. Suppose, for example, that an expression $\forall\mu A(\mu)$ is multiplied first such that the result is $\forall\mu_1 A(\mu_1) \wedge \forall\mu_2 A(\mu_2)$. Then, an expression $\forall\nu B(\nu)$ within the scope of $\forall\mu_1$ is multiplied (modifying $A(\mu_1)$ into $A'(\mu_1)$), which causes $\forall\mu_2 A(\mu_2)$ to need to be multiplied. In this situation, $\forall\mu_2 A(\mu_2)$ is no longer isomorphic to $\forall\mu_1 A'(\mu_1)$ because the scope $A'(\mu_1)$ contains one more multiplied expression than $A(\mu_2)$ does. Therefore, any multiplication of $\forall\mu_2$ that necessitates a further multiplication of $\forall\mu_1$ cannot necessitate an isomorphic $\wedge$I application of $\forall\mu_1$ because the numbers of literals in $A(\mu_2)$ and $A'(\mu_1)$ are no longer identical. However, the following lemma holds:

Lemma 6.17. *The process of successively multiplying expressions necessarily produces isomorphic multiplied expressions $\forall\rho_1 A(\rho_1)$ and $\forall\rho_2 A(\rho_2)$, where the scopes $A(\rho_1)$ and $A(\rho_2)$ each contain the same number of literals.*

Proof. Non-isomorphic multiplied expressions $\forall\mu_1 A'(\mu_1)$ and $\forall\mu_2 A(\mu_2)$ can occur only as a consequence of prior multiplications of quantifiers $\forall\nu$ within at least one of the scopes $A(\mu_1)$ and $A(\mu_2)$. However, the isomorphism of (i) multiplied universally quantified expressions of inner universally quantified expressions separated by existential quantifiers or conjunctions from $\forall\nu$ and (ii) outer universal quantifiers separated by existential quantifiers or conjunctions from $\forall\mu_1$ and $\forall\mu_2$ is not affected. Due to the finite length of D_i and the finite number of nested quantifiers within D_i, any sequence of successive multiplications of universally quantified expressions must, at some point, repeat the multiplication of such *inner* or *outer* universally quantified expressions. Therefore, any non-isomorphic sequence of successive multiplications of universally quantified expressions necessarily produces the repeated multiplication of isomorphic expressions with the same number of literals. □

Theorem 6.18. `T1` *and* `T2` *are sufficient to ensure that the search for a $\wedge$I-minimal proof of D_i in step 2 of the FOL-Decider terminates.*

Proof. In step 2 of the FOL-Decider, a search tree is generated in which each $\wedge$I application corresponds to a branching point. Subsequent to a $\wedge$I application, only a finite number of branches (= alternative proof paths) can be generated. This is so because the alternative proof paths depend only on differences in `L`, $\wp$, and `sub`, which, in turn, all depend on the finite length of D_i^* and are therefore finite. Due to the *Fundamental Principle* `FP`, every conjunct is maintained once selected, and thus, the number of selected conjuncts increases with each $\wedge$I application. Therefore, new branches at the end of the search tree differ from previous branching points on the same proof path by their increased number of selected conjuncts, and the search tree terminates if it terminates in depth.

That the search tree terminates in depth is ensured by the criteria `T1` and `T2`, which are related to the loop list. Any loop list is finite because only a finite number of x variables can be replaced with only a finite number of y variables in only a finite number of pairs of literals due to the finite length of D_i^*. If the initial loop list prior to any $\wedge$I application is empty, then the search for a proof terminates due to `T1` in stage 1. If the multiplication of an x variable μ does not necessitate any further multiplication of an x variable, then the length of the loop list decreases by at least 1 because at least the μ partial list is deleted (cf. Remark 6.12).

As explained in Remark 6.13, subsequent to a $\wedge$I application, either the length of the loop list decreases by 1 or an increase in body length occurs. Thus, as long as the loop list is not empty, and `T1` therefore does not apply, the lengths of the bodies (= numbers of heads

in the bodies) in the loop list increase over consecutive ∧I applications. Consequently, the condition of the *Loop Criterion* must be satisfied after a finite number of steps given Lemma 6.17. This is so because the number of non-isomorphic heads with the same types and numbers of literals from isomorphic scopes is finite due to the fact that the identification of isomorphic heads reduces the index depth to ≤ 2. Therefore, since the process of successive multiplication of universally quantified expressions necessarily results in the multiplication of isomorphic expressions (Lemma 6.17), the substitutions of pairs of literals containing the literals from the scope of the multiplied universal quantifier will, at some stage, be isomorphic. At this point, the ∧I application will be identified by the *Loop Criterion* as a superfluous repetition in the proof search.

The *Loop Criterion* must be satisfied after a finite number of steps even in the case that the y variables `yvars` contain several derivates of the same y variable. This is true because the number of possible substitutions of an x variable μ is bounded by its occurrences in positions in different literals in the (finite) scope $A(\mu)$ in $\forall\mu A(\mu)$. Isomorphic heads contain identical numbers of these positions; thus, there is only a finite number of possible isomorphic heads due to the finite length of isomorphic multiplications of $\forall\mu A(\mu)$.

Therefore, some head in the body of some partial list will become isomorphic to the head of that partial list during the course of ∧I applications unless the loop list is emptied. □

Example 6.19. The following formula (6.1) is a simple example of a formula that has only infinite models. However, there is no need to refer to model theory to prove its non-refutability (satisfiability). It can be proven to be non-refutable on the basis of the ∧I-minimal proof strategy and the *Loop Criterion*.

Let D_i be given as follows:

$$\forall x_1 \exists y_1 (Fx_1y_1 \wedge \forall x_3 (Fx_3y_1 \vee \neg Fx_3x_1)) \wedge \forall x_2 \neg Fx_2x_2 \tag{6.1}$$

(6.1) is equivalent to a formula presented by [Boerger et al.(2001)], p. 33, as an example of a formula that has only infinite models. It is not decided in step 1 of the FOL-Decider. Step 2 starts with the generation of `L`, $\wp$, and `sub` prior to any ∧I application. There is only one initial minimal set of pairs of literals:[22]

$$\texttt{L}: \quad \{\{Fx_1y_1, \neg Fx_3x_1\}, \{Fx_3y_1, \neg Fx_2x_2\}\} \tag{6.2}$$

There is also only one optimized prenex (y_0 precedes any prenex by definition):

$$\wp: \quad \{y_0, x_1, y_1, x_2, x_3\} \tag{6.3}$$

In addition, there is only one `sub` that specifies how to unify (6.2) in relation to (6.3). x_2 and x_3 must both be replaced with y_1 in the second pair of literals $\{Fx_3y_1, \neg Fx_2x_2\}$. x_1 must be replaced with y_1 in the second position in $\neg Fx_3x_1$ to unify the first pair of literals $\{Fx_1y_1, \neg Fx_3x_1\}$. x_1 and x_3 must both be replaced with the same y variable in the first positions in the literals. Thus, σ is unambiguous as long as $\texttt{P}_\wp$, and consequently (6.3), is not considered. However, condition $\texttt{P}_\wp$ on p. 19 is satisfied only if x_1 is replaced with y_0 in addition to y_1 since y_1 is to the right of x_1 in (6.3). Thus, x_1 must be multiplied to replace x_{1_2} with y_{1_1} subsequent to the ∧I application. In addition to the mentioned occurrence of x_1 in the second position in $\neg Fx_3x_1$, x_1 occurs only in the first position in Fx_1y_1 in the

[22] Fx_1y_1 and $\neg Fx_2x_2$ do not constitute a unifiable pair of literals since $\forall x_1 \exists y_1 Fx_1y_1 \wedge \forall x_2 \neg Fx_2x_2$ is not refutable.

first pair of literals $\{Fx_1y_1, \neg Fx_3x_1\}$ in (6.2). At this position, x_1 must be replaced with y_0 because a pair of literals in which both literals contain an x variable `xvar` is unifiable only if in at least one of the two literals, `xvar` must be replaced with a y variable that is to the left of `xvar` in $\wp$.[23] As a consequence of replacing x_1 in the first position in Fx_1y_1 with y_0, x_3 must also be replaced with y_0 in the first position in $\neg Fx_3x_1$. Thus, the following `sub/L` is obtained:

$$\{\{\{\{\{x_1, y_1\}\}, \{\{x_1, x_3, y_0\}\}, \{\}\}, \{Fx_1y_1, \neg Fx_3x_1\}\}, \tag{6.4}$$
$$\{\{\{\{x_2, y_1\}, \{x_3, y_1\}\}, \{\}, \{\}\}, \{Fx_3y_1, \neg Fx_2x_2\}\}\} \tag{6.5}$$

This results in the following substitution list σ:

$$\{\{x_1, y_0, y_1\}, \{x_2, y_1\}, \{x_3, y_0, y_1\}\} \tag{6.6}$$

Since x_1 as well as x_3 must be replaced with y_0 and y_1, the initial loop list contains two partial lists, (6.7) and (6.8). Since this loop list is the initial one, the bodies of the partial lists are empty. I highlight the first lists in boldface:

$$\{\{\{\mathbf{x_1}, \mathbf{y_0}, \mathbf{y_1}\}, \{\{\{\{x_1, y_1\}\}, \{\{x_1, x_3, y_0\}\}, \{\}\}, \{Fx_1y_1, \neg Fx_3x_1\}\},$$
$$\{\{\{\{x_2, y_1\}, \{x_3, y_1\}\}, \{\}, \{\}\}, \{Fx_3y_1, \neg Fx_2, x_2\}\}\}, \{\}\}, \tag{6.7}$$
$$\{\{\mathbf{x_3}, \mathbf{y_0}, \mathbf{y_1}\}, \{\{\{\{x_1, y_1\}\}, \{\{x_1, x_3, y_0\}\}, \{\}\}, \{Fx_1y_1, \neg Fx_3, x_1\}\},$$
$$\{\{\{\{x_2, y_1\}, \{x_3, y_1\}\}, \{\}, \{\}\}, \{Fx_3, y_1, \neg Fx_2, x_2\}\}\}, \{\}\}\} \tag{6.8}$$

Since x_1 as well as x_3 must be replaced with more than one y variable, $\wedge$I must be applied to multiply x_1 and x_3. Both are necessary to unify all pairs of literals from (6.2). One could consider the multiplication of different universally quantified expressions in different orders on different proof paths. However, this would be ineffective. Because the x variables from lists of lengths > 2 in σ must be multiplied anyway, one of the x variables to be multiplied can be selected arbitrarily. If a $\wedge$I-minimal proof exists, it will be found with any order of multiplication of the universally quantified expressions. Thus, the following principle can be established:

Principle of Commutativity (`PC`). *If several x variables must be multiplied, any of them can be multiplied first.*

However, this does not mean that the number of $\wedge$I applications on a proof path and the number of alternative proof paths do not depend on the order in which universally quantified expressions are multiplied. Not the correctness but the efficiency of the $\wedge$I-minimal proof strategy depends on which universally quantified expression is selected to be multiplied first. Therefore, the proof search depends on the implemented strategy for ordering the necessary $\wedge$I applications. In most cases, the FOL-Decider multiplies the outermost universally quantified expressions before the innermost ones because inner expressions are also multiplied in the process of multiplying outer ones. However, the multiplication of existentially quantified expressions increases the complexity of the proof search. For this reason, universally quantified expressions that do not contain existentially quantified expressions

[23] This condition is similar to C3U1 from [Lampert (2019)], p. 28, and is implemented in step 2 of the FOL-Decider.

are multiplied first. Consequently, $\forall x_3(Fx_3y_1 \vee \neg Fx_3x_1)$ is multiplied first in (6.1). This results in the following D_i^*:

$$\forall x_1 \exists y_1(Fx_1y_1 \wedge \forall x_{3_1}(Fx_{3_1}y_1 \vee \neg Fx_{3_1}x_1) \wedge \forall x_{3_2}(Fx_{3_2}y_1 \vee \neg Fx_{3_2}x_1)) \wedge \forall x_2 \neg Fx_2x_2 \tag{6.9}$$

The resulting $\mathtt{sub}'/\mathtt{L}'$ is as follows:

$$\{\{\{\{x_1, y_1\}\}, \{\{x_1, x_{3_1}, y_0\}\}, \{\}\}, \{Fx_1y_1, \neg Fx_{3_1}x_1\}\}, \tag{6.10}$$

$$\{\{\{\{x_2, y_1\}, \{x_{3_2}, y_1\}\}, \{\}, \{\}\}, \{Fx_{3_2}y_1, \neg Fx_2x_2\}\}\} \tag{6.11}$$

Finally, the resulting extended optimized prenex $\wp'$ is

$$\wp' : \quad \{y_0, x_1, y_1, x_2, x_{3_1}, x_{3_2}\} \tag{6.12}$$

Since the multiplied universally quantified expression contains $\vee$, $\mathtt{L}'$ must be extended. There is only one possible extension that satisfies $\bot\mathtt{P}$, $\mathtt{M_L P}$ and PEL, namely, the addition of the unifiable pair of literals $\{Fx_{3_1}y_1, \neg Fx_{3_2}x_1\}$. Since (6.10) and (6.11) must be maintained, according to PE, there are exactly two alternatives for specifying the substitutions of this pair of literals:

$$\textit{Alternative}\,1: \quad \{\{\{\{x_1, y_1\}\}, \{\{x_{3_1}, x_{3_2}, y_1\}\}, \{\}\}, \{Fx_{3_1}y_1, \neg Fx_{3_2}x_1\}\} \tag{6.13}$$

$$\textit{Alternative}\,2: \quad \{\{\{\{x_1, y_1\}\}, \{\{x_{3_1}, x_{3_2}, y_0\}\}, \{\}\}, \{Fx_{3_1}y_1, \neg Fx_{3_2}x_1\}\} \tag{6.14}$$

Alternative 1 is realized on proof path 1, and *Alternative 2* is realized on an alternative proof path 2. In *Alternative 1*, x_{3_1} must be replaced with y_1 in addition to y_0 in (6.10). In *Alternative 2*, x_{3_2} must be replaced with y_0 in addition to y_1 in (6.11). If one counts not merely the $\wedge$I applications on a single proof path but the absolute number of $\wedge$I applications, then the multiplication of x_{3_1} is the second $\wedge$I application, and the multiplication of x_{3_2} is the third. In both cases, applying $\wedge$I results in two further proof paths. In the following, I consider *Alternative 1* only on proof path 1; proof path 2 terminates similarly within the same number of steps.

The modified loop list for *Alternative 1* contains two partial lists, (6.15) and (6.16), since x_1 and x_{3_1} must be multiplied. I highlight the first lists in the head and in the body of the partial lists in boldface:

$$\begin{gathered}\{\{\{\mathbf{x_1}, \mathbf{y_0}, \mathbf{y_1}\}, \{\{\{\{x_1, y_1\}\}, \{\{x_1, x_{3_1}, y_0\}\}, \{\}\}, \{Fx_1y_1, \neg Fx_{3_1}x_1\}\}, \\ \{\{\{x_1, y_1\}\}, \{\{x_{3_1}, x_{3_2}, y_1\}\}, \{\}\}, \{Fx_{3_1}y_1, \neg Fx_{3_2}x_1\}\}, \\ \{\{\{x_2, y_1\}, \{x_{3_2}, y_1\}\}, \{\}, \{\}\}, \{Fx_{3_2}y_1, \neg Fx_2, x_2\}\}\}\}, \{\}\},\end{gathered} \tag{6.15}$$

$$\begin{gathered}\{\{\{\mathbf{x_{3_1}}, \mathbf{y_0}, \mathbf{y_1}\}, \{\{\{\{x_1, y_1\}\}, \{\{x_1, x_{3_1}, y_0\}\}, \{\}\}, \{Fx_1y_1, \neg Fx_{3_1}x_1\}\}, \\ \{\{\{x_1, y_1\}\}, \{\{x_{3_1}, x_{3_2}, y_1\}\}, \{\}\}, \{Fx_{3_1}y_1, \neg Fx_{3_2}x_1\}\}\}\},\end{gathered} \tag{6.16}$$

$$\begin{gathered}\{\{\{\mathbf{x_3}, \mathbf{y_0}, \mathbf{y_1}\}, \{\{\{\{x_1, y_1\}\}, \{\{x_1, x_3, y_0\}\}, \{\}\}, \{Fx_1y_1, \neg Fx_3x_1\}\}, \\ \{\{\{x_2, y_1\}, \{x_3, y_1\}\}, \{\}, \{\}\}, \{Fx_3y_1, \neg Fx_2x_2\}\}\}\}\}\}\end{gathered}$$

The body of partial list (6.15) is still empty. In the case of partial list (6.16), the *Loop Criterion* is not yet satisfied because the second pairs of literals in the head and the list in the body are not isomorphic.

Although x_1 is still to be replaced with y_0 and y_1, derivates of x_3 are to be multiplied prior to (derivates of) x_1, according to the *Commutative Principle* PC (cf. p. 35). Multiplying x_{3_1} results in the following D_i^*:

$$\begin{gathered}\forall x_1 \exists y_1 (Fx_1y_1 \wedge \\ \forall x_{3_{1_1}}(Fx_{3_{1_1}}y_1 \vee \neg Fx_{3_{1_1}}x_1) \ \wedge \forall x_{3_{1_2}}(Fx_{3_{1_2}}y_1 \vee \neg Fx_{3_1}x_1) \wedge \\ \forall x_{3_2}(Fx_{3_2}y_1 \vee \neg Fx_{3_2}x_1)) \wedge \forall x_2 \neg Fx_2x_2\end{gathered} \tag{6.17}$$

The resulting $\mathtt{sub}'/\mathtt{L}'$ is as follows:

$$\{\{\{\{x_1, y_1\}\}, \{\{x_1, x_{3_{1_1}}, y_0\}\}, \{\}\}, \{Fx_1y_1, \neg Fx_{3_{1_1}}x_1\}\}, \tag{6.18}$$

$$\{\{\{\{x_2, y_1\}, \{x_{3_2}, y_1\}\}, \{\}, \{\}\}, \{Fx_{3_2}y_1, \neg Fx_2x_2\}\}\}, \tag{6.19}$$

$$\{\{\{\{x_1, y_1\}\}, \{\{x_{3_{1_2}}, x_{3_2}, y_1\}\}, \{\}\}, \{Fx_{3_{1_2}}y_1, \neg Fx_{3_2}x_1\}\}\}\} \tag{6.20}$$

Finally, the resulting extended optimized prenex $\wp'$ is

$$\wp' : \quad \{y_0, x_1, y_1, x_2, x_{3_{1_1}}, x_{3_{1_2}}, x_{3_2}\} \tag{6.21}$$

Again, $\mathtt{L}'$ must be extended, and there is only one possible extension that satisfies $\perp\mathtt{P}$, $\mathtt{M_L P}$ and PEL, namely, the addition of the unifiable pair of literals $\{Fx_{3_{1_1}}y_1, \neg Fx_{3_{1_2}}x_1\}$. As before, there are two and only two alternatives for specifying the substitutions of this pair of literals:

$$\textit{Alternative}\,1' : \quad \{\{\{\{x_1, y_1\}\}, \{\{x_{3_{1_1}}, x_{3_{1_2}}, y_1\}\}, \{\}\}, \{Fx_{3_{1_1}}y_1, \neg Fx_{3_{1_2}}x_1\}\} \tag{6.22}$$

$$\textit{Alternative}\,2' : \quad \{\{\{\{x_1, y_1\}\}, \{\{x_{3_{1_1}}x_{3_{1_2}}, y_0\}\}, \{\}\}, \{Fx_{3_{1_1}}y_1, \neg Fx_{3_{1_2}}x_1\}\} \tag{6.23}$$

In the case of *Alternative 1'*, $x_{3_{1_1}}$ must again be replaced with y_0 and y_1 to unify the pairs of literals in (6.18) and (6.22), which are isomorphic to (6.10) and (6.13). Therefore, the loop list satisfies the *Loop Criterion* and terminates the proof path that extends (6.18) - (6.20) by adding (6.22). Since the x_1 partial list is irrelevant for the application of the *Loop Criterion*, I quote only the $x_{3_{1_1}}$ partial list:

$$\begin{gathered}\{\{\{\mathbf{x_{3_{1_1}}}, \mathbf{y_0}, \mathbf{y_1}\}, \{\{\{\{x_1, y_1\}\}, \{\{x_1, x_{3_{1_1}}, y_0\}\}, \{\}\}, \{Fx_1y_1, \neg Fx_{3_{1_1}}x_1\}\}, \\ \{\{\{\{x_1, y_1\}\}, \{\{x_{3_{1_1}}, x_{3_2}, y_1\}\}, \{\}\}, \{Fx_{3_{1_1}}y_1, \neg Fx_{3_2}x_1\}\}\}\}, \\ \{\{\{\mathbf{x_3}, \mathbf{y_0}, \mathbf{y_1}\}, \{\{\{\{x_1, y_1\}\}, \{\{x_1, x_3, y_0\}\}, \{\}\}, \{\{Fx_1y_1, \neg Fx_3x_1\}\}, \\ \{\{\{\{x_2, y_1\}, \{x_3, y_1\}\}, \{\}, \{\}\}, \{Fx_3y_1, \neg Fx_2x_2\}\}\}\}, \\ \{\{\mathbf{x_{3_1}}, \mathbf{y_0}, \mathbf{y_1}\}, \{\{\{\{x_1, y_1\}\}, \{\{x_1, x_{3_1}, y_0\}\}, \{\}\}, \{Fx_1y_1, \neg Fx_{3_1}x_1\}\}, \\ \{\{\{\{x_1, y_1\}\}, \{\{x_{3_1}, x_{3_2}, y_1\}\}, \{\}\}, \{Fx_{3_1}y_1, \neg Fx_{3_2}x_1\}\}\}\}\}\end{gathered} \tag{6.24}$$

The last list in the body is isomorphic to the head of (6.24).

The resulting $\mathtt{sub}/\mathtt{L}$ on the proof path that consists of (6.18) - (6.20) and (6.23) (= *Alternative 2'*) is not eliminated by the *Loop Criterion* T2. This is so because the pairs of

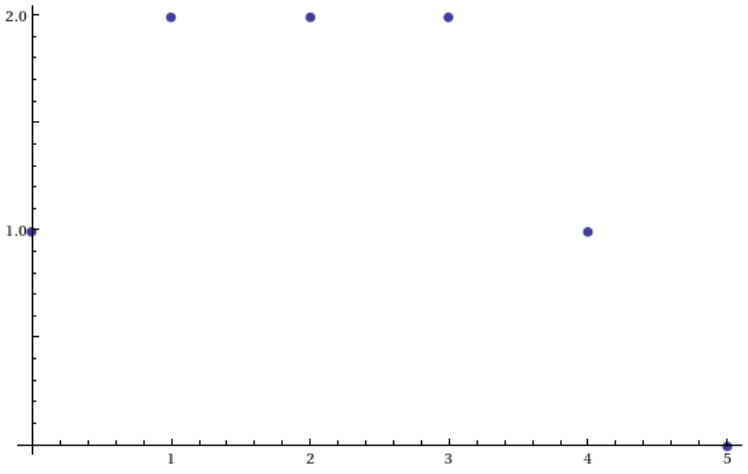

Figure 2: Number of proof paths vs. number of proof steps in the `sat`-proof of (6.1)

literals in (6.20) and (6.23) that contain literals in the scope of $\forall x_{3_{1_2}}$ are not isomorphic to the pairs of literals in (6.10) and (6.13) that contain literals selected from the scope of $\forall x_{3_1}$ in the previous `sub`/`L` resulting from the first $\wedge$I application.

Regarding the `sub`/`L` resulting from *Alternative 2'*, $x_{3_{1_2}}$ must be replaced with y_0 and y_1. The multiplication of $x_{3_{1_2}}$ (= the fourth $\wedge$I application in the proof search) results in two alternative sets of $\mathtt{sub}''/\mathtt{L}''/\wp'$. The modification of the loop list in accordance with these $\mathtt{sub}''/\mathtt{L}''/\wp$ alternatives satisfies the conditions of the *Loop Criterion* in both cases. In one case, $x_{3_{1_{2_1}}}$ must be replaced with y_0 and y_1; in the other, $x_{3_{1_{2_2}}}$ must be replaced with y_0 and y_1; and in both cases, the pairs of literals and their substitutions are isomorphic to those of (6.20) and (6.23) (*Alternative 2'*).

The multiplication of x_{3_2} (= the third $\wedge$I application in the proof search) yields results similar to those of the multiplication of x_{3_1}; the multiplication of $x_{3_{2_1}}$ (= the fifth $\wedge$I application in the proof search) subsequent to the multiplication of x_{3_2} yields results similar to those of the multiplication of $x_{3_{1_2}}$. Thus, the proof search terminates after 5 steps due to `T2`. Figure 2 depicts the relation between the number of proof steps or $\wedge$I applications (x axis) and the number of proof paths (y axis). The number of proof paths can be reduced by only 1 in one proof step. In a typical `sat`-proof, the number of proof paths initially increases rapidly and then, at a certain point, begins to slowly decrease. By contrast, `False`-proofs terminate immediately at any step and any number of proof paths.

Remark 6.20. The D_i given in (6.25) is equivalent to a formula from [Dreben (1979)], p. 120, which is another formula that has only infinite models:

$$\forall x_1 \exists y_1 (\forall x_4 (\neg P x_1 x_4 \vee Q y_1 x_4) \wedge \forall x_5 (\neg Q x_1 x_5 \vee Q y_1 x_5)) \wedge \forall x_2 \neg Q x_2 x_2 \wedge \forall x_3 P x_3 x_3 \quad (6.25)$$

Like (6.1), (6.25) can be rather rapidly proven by the FOL-Decider to be non-refutable (cf. figure 3).

(6.1) (= SYO637+1.p in the TPTP library) and (6.25) (= SYO635+1.p in the TPTP library) are examples of formulas with only infinite models for which the FOL-Decider

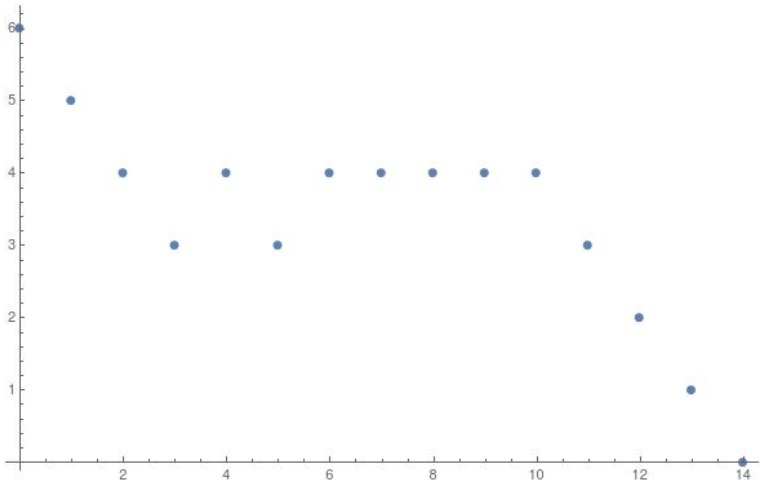

Figure 3: Number of proof paths vs. number of proof steps in the `sat`-proof of (6.25)

reaches a decision, whereas these formulas are not decided by other logic engines, such as Beagle, Darwin, i-Prover, E, Vampire and SPASS.

7. Output

If the FOL-Decider finds a ∧I-minimal proof for D_i in step 2, then the output, in addition to D_i, is as follows:

(1) the rectified and purged* anti-prenex normal form D_i^*,
(2) the minimal set `L` of unifiable pairs of literals,
(3) the unambiguous substitution list σ that specifies how the x variables must be replaced with y variables such that all pairs of literals from `L` are unified, and
(4) an optimal prenex $\wp$ that allows the substitutions in σ to be performed by applying ∀E.

Thus, a recipe for a proof of contradiction is provided. From D_i^*, one can read off which universally quantified expressions in D_i must be multiplied and which conjuncts can be deleted; $\wp$ identifies the prenex normal form D_i^{**} that is to be generated from D_i^* by applying PN laws; σ determines how ∀E must be applied to derive an explicit contradiction D_i^{***} from D_i^{**}; and `L` specifies the pairs of literals that must be unified such that each disjunct of the DNF matrix of D_i^{***} contains an explicit contradiction. For an example, cf. Example 5.28, p. 24, with D_i = (5.42), D_i^* = (5.46), `L` = (5.47), σ = (5.48), and $\wp$ = (5.49).

The FOL-Decider ultimately returns `False` if all disjuncts D_i of the FOLDNF resulting from step 1 of the FOL-Decider are proven to be refutable. In this case, the user receives the FOLDNF that is sat-equivalent to the input formula ϕ and all recipes for the ∧I-minimal proofs for the disjuncts D_i of the FOLDNF.

Once the proof search for a D_i terminates without a ∧I-minimal proof having been found, the FOL-Decider returns `sat`. In this case, D_i and, therefore, the FOLDNF and the input formula ϕ are proven to be non-refutable. The user can ask for details of the `sat`-proofs, e.g., diagrams such as figures 2 and 3; the D_i^*, `sub`/`L`, and $\wp$ for each proof step; or the applications of the various principles of the ∧I-minimal proof strategy.

8. Effective Proof Search

The search for ∧I-minimal proofs can be optimized from several perspectives. Three optimization strategies are already implemented as optional tools. First, one can minimize the FOLDNFs in step 1 of the FOL-Decider using the algorithm described in [Lampert (2017)]. Furthermore, the scopes of existential quantifiers can be further minimized in step 1 of the FOL-Decider through ∃M optimization (cf. section 7 of [Lampert (2019)] and footnote 9). However, both of these tools sometimes speed up the evaluation but sometimes slow it down. Finally, one can also apply an optional model-theoretic tool to search for a model up to an upper bound independent of the proof-theoretic strategy of the FOL-Decider. In addition, several optional tools can be applied to provide models in the case of a `sat`-proof. However, the decision procedure based on the ∧I-minimal strategy is independent of model theory. If one intends to understand this strategy, one should abstain from invoking model theory.

Three further optimization strategies have not yet been implemented. First, the combinatorial generation of $\mathtt{sub}/\mathtt{L}/\wp$ alternatives and the *Loop Criterion* may be restricted by further criteria. Second, the FOL-Decider does not currently apply any principles that rely on comparing alternative proof paths, apart from deleting duplicates. Third, one could classify patterns of $\mathtt{sub}/\mathtt{L}/\wp$ for the termination of proof paths. Furthermore, many improvements to both the concrete implementation and the proof search strategy are possible. However, the FOL-Decider is not designed to implement a fast and effective proof search. In this respect, other logic engines are preferable. Instead, the intent in developing the FOL-Decider was to implement the ∧I-minimal proof strategy to demonstrate its feasibility.

If one is looking for a transparent and elegant proof search that implements the logical ideas underlying the ∧I-minimal strategy, then principles and criteria that allow one to terminate proof paths as soon as possible are indispensable. Any optimization, however, is prone to error. For this reason, I refer to a "maximal search tree" instead of an "effective search tree" in the following.

Definition 8.1. A *maximal search tree* generates all combinatorially possible non-duplicate alternatives of the following:

(1) D_i^* resulting from different selections of the order in which to multiply universally quantified expressions,
(2) minimal and non-minimal sets $\mathtt{L}^*$ of unifiable pairs of literals,
(3) optimized and not optimized (logically valid) prenexes $\wp^*$, and
(4) all alternative specifications $\mathtt{sub}^*$ of substitutions of x variables in xx positions with y variables for unification

such that the principles `FP`, $\bot\mathtt{P}$, and $\mathtt{P}_\wp$ and the termination criteria `T1` and `T2` are satisfied.

Remark 8.2. I abstain here from providing an exact algorithmic definition of "all combinatorial alternatives of D_i^*, $\mathtt{L}^*$, $\wp^*$, and $\mathtt{sub}^*$ in the course of each ∧I application". Likewise, I abstain from specifying formulas for calculating the number of these alternatives. It is sufficient to recognize that they are finite due to the finite numbers of literals, variables and universal quantifiers in each step on each proof path. The algorithmic specification of `FP`, $\bot\mathtt{P}$, and $\mathtt{P}_\wp$ for a ∧I-minimal proof search is also trivial. Thus, it should be accepted that a maximal search tree for D_i can be generated.

Remark 8.3. A maximal search tree contains all proof paths that satisfy the principles $\mathtt{M_LP}$, PE, and PC, but it is not restricted to these proof paths. FP, $\mathtt{P}_\wp$, and $\bot\mathtt{P}$ are *necessary principles* for a $\wedge$I-minimal proof search, and T1 and T2 are *necessary* criteria for the termination of a proof search. By contrast, $\mathtt{M_LP}$, PE, and PC are principles of an *effective* search strategy for $\wedge$I-minimal proofs.

Remark 8.4. The proof of Theorem 6.18 does not depend on the efficiency of a $\wedge$I-minimal proof search. As long as the loop list is not empty, the body lengths increase with every $\wedge$I application that does not decrease the length of the loop list. Consequently, the head of some partial list will become isomorphic to some head in the body of that partial list within a finite number of steps because only a finite number of isomorphic heads can be generated from D_i.

Theorem 8.5. *A maximal search tree for a D_i is finite.*

Proof. The proof of this theorem follows from Theorem 6.18 and Remark 8.4 and can be summarized as follows:

(1) The combinatorial alternatives of D_i^*, $\mathtt{L}^*$, $\wp^*$ and $\mathtt{sub}^*$ in the course of each $\wedge$I application are finite. Therefore, each branching point is finite.
(2) If the search for a proof along a proof path does not terminate due to an empty loop list or the necessary principles for the termination of a $\wedge$I-minimal proof search, then the proof search will eventually lead to an isomorphic $\wedge$I application due to the finite length of D_i. Therefore, T2 applies unless a proof path is terminated due to the necessary principles T1 or FP, $\bot\mathtt{P}$ and $\mathtt{P}_\wp$.

□

9. Correctness

Proving the correctness of the FOL-Decider requires proving that the FOL-Decider returns False iff the input formula ϕ is refutable. The proof in the direction from left to right is easy.

Theorem 9.1. *If the FOL-Decider returns* False*, then the input formula ϕ is refutable.*

Proof. The result of step 1 of the FOL-Decider is an FOLDNF. According to Theorem 2.8, the input formula ϕ is sat-equivalent to its FOLDNF. If the FOL-Decider returns False as a result in step 1, then the simple False/sat check applied in step 1 and referred to in the proof of Theorem 2.8 already identifies the FOLDNF and, therefore, the sat-equivalent initial formula ϕ as refutable. Otherwise, the FOL-Decider returns False iff it returns False for each disjunct D_i from disjuncts D_1 to D_n of the FOLDNF. Since ϕ is refutable iff each disjunct D_i is refutable, ϕ is refutable if each D_i is indeed refutable in the case that the FOL-Decider returns False for each D_i as a result of step 2. If the FOL-Decider returns False for D_i as a result of step 2, it returns a recipe for a proof of contradiction for D_i within the NNF-calculus. This recipe determines the necessary applications of $\wedge$I, an optimal prenex that can be generated via PN laws and the applications of $\forall$E that are needed to derive an explicit contradiction. From the correctness of the NNF-calculus (cf. Theorem 3.3 in [Lampert (2019)]), it follows that each D_i and, therefore, the FOLDNF and ϕ are indeed refutable. □

For a complete and detailed proof in the direction from right to left, it is necessary to replace the definitions and principles serving as the basis of the ∧I-minimal proof search strategy presented in this paper with their exact algorithmic implementations. In particular, this concerns the computations of `L`, $\wp$, and `sub` and their modification and extensions. However, I abstain from this here. It is not necessary to describe the algorithmic details if what is in question is not the correctness of the specific program (the FOL-Decider) but rather the correctness of its underlying decision strategy. To prove the decidability of D_i, it is sufficient to prove that the finite *maximal* search tree for ∧I-minimal proofs in the NNF-calculus contains a ∧I-minimal proof if D_i is refutable.

Definition 9.2. A maximal search tree *contains* a ∧I-minimal proof iff it generates a combination of D_i^*, `L`, $\wp$, and `sub` that constitutes a ∧I-minimal proof.

Lemma 9.3. *A maximal search tree contains all ∧I-minimal proofs.*

Proof. According to Definition 8.1, a maximal search tree generates all combinatorially possible D_i^*, $\texttt{L}^*$, $\wp^*$, and $\texttt{sub}^*$. By definition, this totality of combinations contains all combinations of D_i^*, `L`, $\wp$, and `sub` that constitute ∧I-minimal proofs (cf. Remark 8.3). □

Referring to a maximal proof tree circumvents the necessity of proving the correctness of the principles $\texttt{M}_\texttt{L}\texttt{P}$, `PE`, and `PC` of an effective search strategy for ∧I-minimal proofs. The decidability of FOL through the ∧I-minimal proof strategy in the NNF-calculus can be proven on the basis of Lemma 9.3 without considering the details of its implementation in the FOL-Decider.

Theorem 9.4. FOL *is decidable.*

Proof. According to Theorem 2.8, ϕ is sat-equivalent to its FOLDNF. Thus, ϕ is decidable if D_i is decidable. That D_i is decidable follows from Theorem 8.5 (finiteness of the maximal search tree), Lemma 9.3 (completeness of a maximal search tree with respect to ∧I-minimal proofs), Theorem 5.2 (completeness of ∧I-minimal proofs of D_i in the complete NNF-calculus), Theorem 9.1 (correctness of `False` outputs) and Theorem 6.16 (correctness of the *Loop Criterion* for the ultimate termination of proof paths). □

Remark 9.5. Theorem 9.4 contradicts the meta-mathematical proof of the Church-Turing theorem. In contrast to this meta-mathematical proof, the proof of FOL's decidability rests on nothing but purely logical considerations concerning ∧I-minimal proofs. This proof is independent of the meta-mathematical proof method, which relies on translations of recursive or Turing-computable functions into the language of logic in order to express those computable functions. Given the correctness of the purely logical reasoning concerning the decidability of FOL, this suggests that the translation procedures are not correct in the case of the decision function for FOL-provability, cf. [Lampert (2019)].

Acknowledgements. I am grateful to Anderson Nakano for discussing my algorithm and its proof as well as the critique of the Church-Turing theorem. Furthermore, I am grateful to Karsten Müller, Michael Taktikos, Pietro Fornara, Yvonne Lampert, Markus Säbel, Stefan Steins, Geoff Suttcliffe and Victor Rodych.

References

[Boerger et al.(2001)] Börger, E., Grädel, E. & Gurevich, Y.: *The Classical Decision Problem*, Springer.

[Boolos et al. (2003)] Boolos, G.S., Burgess, J.P., Jeffrey, R.C.: *Computability and Logic*, forth edition, Cambridge: Cambridge University Press.

[Dreben (1979)] Dreben, B., Goldfarb, W.D.: *The Decision Problem. Solvable Classes of Quantificational Formulas*, Addison-Wesley, London.

[Hilbert(1970)] Hilbert, D., Bernays, P.: *Grundlagen der Mathematik*, Band 1, Springer, Berlin.

[Lampert (2017)] Lampert, T. 2017: "Minimizing disjunctive normal forms of pure first-order logic", *The logic Journal of the IGPL*, 25 (3), 325-347.

[Lampert (2019)] Lampert, T. 2019: "A Decision Procedure for Herbrand Formelae without Skolemization", https://arxiv.org/abs/1709.00191, 1-30.

[Lampert (2019)] Lampert, T. 2019: "The Underdetermination of the Church-Turing Theorem", preprint: http://www2.cms.hu-berlin.de/newlogic/webMathematica/Logic/undecidability_underdetermined.pdf.

[Quine (1982)] Quine, W.V.O.: *Methods of Logic*, Fourth Edition, Harvard University Press, Harvard, MA.