Minimizing Disjunctive Normal Forms of Pure First-Order Logic

Timm Lampert

This is a penultimate draft of a paper published in *Logic Journal of the IGPL 25.3* 2017, p. 325-347

Abstract

In contrast to Hintikka's enormously complex distributive normal forms of first-order logic, this paper shows how to generate minimized disjunctive normal forms of first-order logic. An effective algorithm for this purpose is outlined, and the benefits of using minimized disjunctive normal forms to explain the truth conditions of propositions expressible within pure first-order logic are presented.

keywords. first-order logic, disjunctive normal forms, purification, optimization, Quine-McCluskey algorithm

1 Introduction

Unlike disjunctive normal forms (DNFs) of propositional logic, DNFs of first-order logic (FOLDNFs) have received little attention. Instead, prenex normal forms (PNFs) dominate the *metatheoretic* study of decidable and undecidable fragments of first-order logic (FOL), and clausal normal forms, particularly Skolem normal forms based on conjunctive normal forms (CNFs), dominate the *computational* search for effective FOL algorithms. Compared with these established normal forms, FOLDNFs have not been studied in the same amount of detail.

Hintikka's distributive normal forms of FOL constitute an important exception to this neglect of FOLDNFs.[1] These normal forms correspond to canonical DNFs of propositional logic (CDNFs). They include an *exhaustive and mutually exclusive* enumeration of disjuncts. Hintikka's definition of his distributive normal forms is rather intricate; cf. Hintikka [6, pp. 52-54], section 3, and in more detail, Nelte [10, pp. 36-65], chapter 3. For the purpose of this paper, it suffices to note that the demand for an exhaustive and mutually exclusive enumeration of disjuncts necessitates an extremely high complexity. The length of each disjunct depends on the lengths of (i) the set of all predicates, (ii) the set of all free individual

[1]Cf. in particular Hintikka [6], which is based on his dissertation, Hintikka [5], which was supervised by Georg Henrik von Wright, who paved the way for Hintikka; cf., e.g., Wright [21] and Wright [22]. Ehrenfeucht and Fraissé (cf., e.g., Ehrenfeucht [3]) as well as Oglesby (cf. Oglesby [12]) also simultaneously worked on distributive normal forms, independently and with different motivations; cf. also Scott [17].

symbols (or names), and (iii) the maximal length of sequences of nested quantifiers (= depth d). Even if one considers only formulas of pure FOL without names and functions, only one binary predicate, and formulas of depth 2, this leads to FOLDNFs with 2^{512} disjuncts, where each disjunct contains 512 conjuncts. Thus, the length of Hintikka's distributive normal form for even the simple formula $\exists x \exists y Fxy$ is 2^{512}. Merely increasing the depth by one already results in $2^{2^{1+2^{35}}}$ possible disjuncts (cf. Nelte [10], section 4.1, who provides detailed formulas for measuring the size of Hintikka's normal forms). Because of this immense complexity, Hintikka's normal forms cannot earn more than theoretical interest.

The reason for this complexity lies inHintikka's intention to use his distributive normal forms in a logical theory of probability and, consequently, in information and game theory.[2] However, this motivation faces severe problems in regards to FOL.

The application of probability to CDNFs is clear-cut in the case of propositional logic: Here, any literal has an a priori probability of $\frac{1}{2}$, and the probabilities of disjuncts and disjunctions can be straightforwardly computed by multiplying and adding the corresponding probabilities. However, things become complicated as soon as FOLDNFs of polyadic FOL are considered. One can no longer presume that conjuncts are logically independent. Hence, it is unclear how to assign probabilities to single conjuncts in Hintikka's normal forms. Furthermore, conjuncts that are logically implied by other conjuncts cannot contribute to the probability of a disjunct, and disjuncts that are inconsistent cannot contribute to the probability of a disjunction.[3] Thus, the application of probability theory in this case is not as clear-cut as in propositional logic, and FOLDNFs cannot be used to compute probabilities given the undecidability of FOL.

However, the interest in FOLDNFs need not be motivated by a logical theory of probability. Hintikka was influenced by von Wright, who pioneered the study of distributive normal forms and supervised Hintikka's dissertation on the topic. Both, in turn, were inspired by Wittgenstein's programmatic ideas on logic in general as well as on a logical theory of probability based on CDNFs in particular. From a Wittgensteinian perspective, one should distinguish a general motivation and a more specific motivation for studying DNFs. First of all, DNFs are a perspicuous way to represent the truth conditions of instances of formulas. DNFs in general provide an alternative to the mathematical model-theoretic approach to semantics. As a consequence of using DNFs to explain the truth conditions of propositions, there is no need to refer to models that rely on set theory and refer to countably or even uncountably infinite domains. I explain this statement in more detail in section 3. However, although von Wright's and Hintikka's study of DNFs is obviously influenced by this general motivation, it is only their specific interest in a logical foundation of probability that necessitates an exhaustive and mutually exclusive enumeration of disjuncts, which, in turn, induces the enormous complexity of their, in a sense, "canonical" disjunctive normal forms of FOL.

Contrary to von Wright and Hintikka, one should seek FOLDNFs of *minimal length* if one is interested in representing the truth conditions of instances of formulas by using DNFs as a perspicuous form of knowledge representation. In fact, there is an alternative, much simpler path leading to FOLDNFs that calls for anti-prenex normal forms that may be written in

[2]Cf. Hintikka [8] in particular; cf. also Hintikka [7].

[3]Nelte [10], section 4.4, defines a lower bound on the fraction of inconsistent disjuncts. Roughly speaking, "nearly all" (p. 86) disjuncts of Hintikka's normal forms are inconsistent as soon as polyadic predicates and overlapping scopes of quantifiers are considered.

terms of CNFs or DNFs. However, this path has, to my knowledge, been neither pursued to generate minimized FOLDNFs nor motivated as an alternative to a traditional semantic approach to FOL. Behmann [1] is known to be the first to have considered anti-prenex normal forms to define a decision procedure for monadic FOL. He also called for extending his investigations to the entire realm of FOL; see Behmann [1, p. 226]. Church [2, p. 58] labels anti-prenex normal forms of monadic FOL "Behmann's normal forms". Hilbert & Bernays [4, p. 145ff.] refer to the process of generating Behmann's normal forms as "analysis to primary formulas". However, although their description of the process is more explicit than Behmann's explanation, they still do not explicitly extend this process to the entire realm of FOL. This is done in the fourth edition of Quine's *Methods of Logic*; see Quine [15], chapter 23. Quine calls the process of generating anti-prenex normal forms "purification" and calls the resulting "primary formulas" "purified formulas".[4] In the following, I will adopt the term "primary formula" from Hilbert and Bernays to refer to conjuncts within disjuncts of FOLDNFs based on anti-prenex normal forms.

In contrast to Hintikka's conjuncts of the disjuncts of his distributive normal forms, primary formulas are not required to satisfy certain combinatoric constraints that depend on the set of predicates as well as on the depth of nested quantifiers and thus induce complexity. Instead, primary formulas can be defined by conditions concerning the quantifiers preceding conjunctions and disjunctions. These conditions constitute a limit on the driving of quantifiers inwards to a maximal extent by means of logical equivalence rules that do not yet involve rules for minimizing formulas. For simplicity, throughout the remainder of this paper, I avoid names from the language of logic and thus restrict the consideration of FOL to that of pure FOL. Consequently, all positions in propositional functions of length ≥ 1 are occupied by variables bound by universal or existential quantifiers. Neither Hilbert & Bernays [4] nor Quine [15] provides a precise definition of primary formulas. The following definition offers a syntactic definition that satisfies the demand to define formulas with quantifiers that are driven inwards to the maximal extent. This definition is based on negation normal forms (NNFs), which contain only $\wedge$ and $\vee$ as dyadic sentential connectives and $\neg$ only directly in front of an atomic propositional function.

Definition 1. (Primary formula)

1. Any 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 of 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 of that sequence.
3. All primary formulas are NNFs that satisfy condition 1 or 2.

Thus, e.g., $\exists x \exists y Fxy$ and $\exists y_1(\exists y_2 Fy_1y_2 \wedge \exists y_3 \forall x_1 \forall x_2(Gx_1x_2y_1 \vee \forall x_3 Hx_1x_2x_3))$ are primary formulas. By contrast, $\exists y_1 \exists y_2(Fy_1y_2 \wedge Gy_2)$ or $\forall x_1 \exists y_1(Fx_1 \vee Gy_1)$ are not primary formulas.

[4]Quine [13] already uses anti-prenex normal forms but in a manner that is similar to von Wright's usage.

$\top$ and $\bot$ are also considered to be NNFs and thus, according to condition 1, represent a special case of primary formulas.

The second conjunct in conditions 2(a) and 2(b) is needed if one intends to refer to formulas with quantifiers driven inwards to the maximal extent. However, as I note on p. 7, neither the procedure suggested by Hilbert and Bernays nor that defined by Quine satisfies the condition that *each* of the existential (universal) quantifiers in the corresponding sequence of quantifiers binds a variable contained in all conjuncts (disjuncts) of the corresponding conjunction (disjunction). Instead, they only require that this be the case for the innermost quantifier of a sequence of existential (universal) quantifiers. However, I will show in the next section how this deficiency can easily be overcome by supplementing their procedures with certain rules for the ordering of quantifiers and their scopes.

Based on Definition 1, FOLDNFs, as used in the following, are defined as follows:

Definition 2. (FOLDNF)
FOLDNFs are disjunctions of conjunctions of primary formulas.

Both disjunctions and conjunctions may have length 1. Thus, the formula $\exists x \exists y Fxy$ discussed above is an FOLDNF, although to satisfy Hintikka's conditions, it must be converted into a disjunction with 2^{512} disjuncts, each containing 512 conjuncts.

Hilbert, Bernays and Quine follow Behmann in considering anti-prenex normal forms essentially in connection with effective decision procedures. Anti-prenex normal forms are truth functions of primary formulas. They need not be further converted into FOLDNFs. In connection with clausal normal forms, for example, the process of "analysis to primary formulas", or "purification" in general, is also known as "miniscoping", cf. Nonnengart & Weidenbach [11]. Although studying anti-prenex normal forms in terms of FOLDNFs is also of interest for investigating decision problems, I abstain from addressing this issue in the following. Instead, it seems to me surprising that the most obvious and intuitive use of DNFs, namely, to explain the conditions for the truth of instances of formulas by finite means and based on an equivalence procedure within logic alone, is not considered in greater detail in connection with the motivation to extend the DNFs of propositional logic to FOLDNFs. Unfortunately, von Wright's and Hintikka's intent is to use FOLDNFs in combination with probability theory, and thus, they do not investigate the far less complex FOLDNFs expressed in terms of DNFs of primary formulas. In the following, I intend to overcome this deficiency.

In contrast to the metatheoretic use of PNFs and the computational use of clausal normal forms (or, likewise, anti-prenex normal forms), FOLDNFs expressed in terms of disjunctions of conjunctions of primary formulas can be utilized for the *logico-philosophical* enterprise of analysing and explaining first-order formulas (or their instances) by converting the initial formulas, mechanically and through equivalence transformation, into a more perspicuous form that allows one to identify the truth conditions of instances of the initial formulas in a plain and neat way. This goal is important for a perspicuous means of *knowledge representation* based on the language of FOL. If one wishes to understand first-order formulas, one must answer the question of what they contribute to the truth conditions of their instances by virtue of their logical form. The best way to answer this question is by referring to their minimized FOLDNFs. Before I address the question of minimizing FOLDNFs in section 4, I elucidate how to explain truth conditions by means of FOLDNFs in section 3. First, however, let us specify the process of purification.

2 Purification

In this section, I describe an algorithm that (i) drives quantifiers inwards to the greatest possible extent and (ii) expands the scopes of the quantifiers to the minimum possible extent while achieving (i).

The process of purification begins from NNFs of FOL. If an initial formula is not an NNF of this sort, it can be converted into such a form in accordance with well-known rules.

Let our initial formula ϕ be

$$\exists y \forall x(\neg(((\neg Fxx \land Gy) \lor Hxy) \to \neg P)) \tag{1}$$

Then, the process of purification starts with the NNF of (1):

$$\exists y \forall x(((\neg Fxx \land Gy) \lor Hxy) \land P) \tag{2}$$

In contrast to prenexing, purification drives quantifiers inwards. This is done by applying PN rules, i.e., the rules used for prenexing, in the opposite direction (= *miniscoping*). Table 1 lists the PN laws. They are applied from left to right to drive quantifiers inwards.

$\forall\nu(A \land B(\nu))$	$\dashv\vdash$	$A \land \forall\nu B(\nu)$	PN1
$\forall\nu(B(\nu) \land A)$	$\dashv\vdash$	$\forall\nu B(\nu) \land A$	PN2
$\forall\nu(A \lor B(\nu))$	$\dashv\vdash$	$A \lor \forall\nu B(\nu)$	PN3
$\forall\nu(B(\nu) \lor A)$	$\dashv\vdash$	$\forall\nu B(\nu) \lor A$	PN4
$\exists\nu(A \land B(\nu))$	$\dashv\vdash$	$A \land \exists\nu B(\nu)$	PN5
$\exists\nu(B(\nu) \land A)$	$\dashv\vdash$	$\exists\nu B(\nu) \land A$	PN6
$\exists\nu(A \lor B(\nu))$	$\dashv\vdash$	$A \lor \exists\nu B(\nu)$	PN7
$\exists\nu(B(\nu) \lor A)$	$\dashv\vdash$	$\exists\nu B(\nu) \lor A$	PN8
$\forall\nu(A(\nu) \land B(\nu))$	$\dashv\vdash$	$\forall\nu A(\nu) \land \forall\nu B(\nu)$	PN9
$\exists\nu(A(\nu) \lor B(\nu))$	$\dashv\vdash$	$\exists\nu A(\nu) \lor \exists\nu B(\nu)$	PN10

Table 1: PN laws applied to drive quantifiers inwards

First, universal quantifiers preceding a conjunction and existential quantifiers preceding a disjunction are driven inwards to the maximal extent by applying PN1 to PN10. (= *1st miniscoping*).

In the case of formula (2), this results in the following formula:

$$\exists y \forall x((\neg Fxx \land Gy) \lor Hxy) \land P \tag{3}$$

To drive quantifiers farther inwards, the quantifiers must first be sorted (= *quantifier sorting*). This is done by first generating sequences of universal (existential) quantifiers of maximal length that precede disjunctions (conjunctions): All universal (existential) quantifiers that are separated only by disjunctions (conjunctions) are grouped together (= *partial prenexing*). This is done by applying PN3 to PN6 from right to left. Renaming of the variables ensures that no two quantifiers bind the same variable.[5] Then, sequences of universal (existential) quantifiers preceding a disjunction (conjunction) are sorted such that universal (existential) quantifiers binding variables that occur in m of the disjuncts (conjuncts) appear to the right of universal (existential) quantifiers binding variables that occur in n of the disjuncts (conjuncts), where $m < n$ (= *prenex sorting*). This is done by applying the following laws:

$\forall\mu\forall\nu A(\mu,\nu)$	$\dashv\vdash$	$\forall\nu\forall\mu A(\mu,\nu)$	$\forall\mathcal{V}$
$\exists\mu\exists\nu A(\mu,\nu)$	$\dashv\vdash$	$\exists\nu\exists\mu A(\mu,\nu)$	$\exists\mathcal{V}$

Table 2: Laws applied for quantifier sorting

Likewise, the disjuncts are sorted such that disjuncts containing a variable μ bound by a universal quantifier $\forall\mu$ that appears to the right of a universal quantifier $\forall\nu$ in the same sequence of universal quantifiers also appear to the right of disjuncts containing ν (and not μ). Thus, if $\forall\mu$ is the innermost quantifier of a sequence of universal quantifiers, then all disjuncts containing μ are selected and placed farthest to the right in the order of the disjuncts in the scope of universal quantifiers, and so on for all other universal quantifiers (= *scope sorting*). The same process is applied to the scopes of each sequence of existential quantifiers with respect to the order of the conjuncts. This process involves the application of well-known rules of propositional logic:

$A \wedge B$	$\dashv\vdash$	$B \wedge A$	KOM$\wedge$
$A \vee B$	$\dashv\vdash$	$B \vee A$	KOM$\vee$
$(A \wedge B) \wedge C$	$\dashv\vdash$	$A \wedge (B \wedge C)$	ASS$\wedge$
$(A \vee B) \vee C$	$\dashv\vdash$	$A \vee (B \vee C)$	ASS$\vee$

Table 3: Laws applied for scope sorting

On the basis of this sorting, universal quantifiers preceding disjunctions and existential quantifiers preceding conjunctions can be driven inwards by again applying the PN laws from left to right (= *2nd miniscoping*).

Neither Hilbert & Bernays [4, pp. 144-148] nor Quine [15, pp. 126-129] considers quantifier sorting. However, the following simple example demonstrates that this procedure is required to drive the quantifiers inwards as far as possible:

$$\exists y_2\exists y_1(\exists y_3(Hy_3 \wedge Ky_3) \wedge Fy_1y_2 \wedge Gy_1) \tag{4}$$

Without quantifier sorting, the scopes of $\exists y_1$ and $\exists y_2$ are not minimized. Thus, according to the algorithm described by Hilbert, Bernays and Quine, (4) is a primary formula, although it does not fully satisfy condition 2 of Definition 1. However, *quantifier* and *scope sorting*

[5]I omit the details of the necessary renaming techniques in the following; for details, cf. Nonnengart & Weidenbach [11, p. 344f.], section 3.4.

results in the following formula:

$$\exists y_3 \exists y_1 \exists y_2 (Hy_3 \wedge Ky_3 \wedge Gy_1 \wedge Fy_1y_2) \tag{5}$$

Applying the PN laws to (5) in a *2nd* process of *miniscoping* results in the following:

$$\exists y_3 (Hy_3 \wedge Ky_3 \wedge \exists y_1 (Gy_1 \wedge \exists y_2 Fy_1y_2) \tag{6}$$

In the case of formula (3), however, no sorting of quantifiers or scopes is needed, as the sequences of existential or universal quantifiers are all of length 1. Consequently, no 2nd miniscoping is applied.

The two miniscoping processes mentioned thus far minimize the scopes of the quantifiers without expanding their scopes a priori. However, for the scopes of the quantifiers to be minimized to the greatest possible extent, the scopes of all universal quantifiers $\forall \nu$ must be converted into CNFs if they are of the form $A(\nu) \vee B(\nu)$, and the scopes of all existential quantifiers $\exists \mu$ must be converted into DNFs if they are of the form $A(\mu) \wedge B(\mu)$ (= *scope conversion*).[6] This is done by applying the distributivity laws of propositional logic:

$(A \vee B) \wedge C$	$\dashv\vdash$	$(A \wedge C) \vee (B \wedge C)$	DIS1
$(A \wedge B) \vee C$	$\dashv\vdash$	$(A \vee C) \wedge (B \vee C)$	DIS2

Table 4: Distributivity laws for converting scopes into CNFs or DNFs

In the case of formula (3), the scope of $\forall x$ must be converted to a CNF by first applying DIS2:

$$\exists y \forall x ((\neg Fxx \vee Hxy) \wedge (Gy \vee Hxy)) \wedge P \tag{7}$$

Hilbert & Bernays [4, pp. 144-148] convert the scopes of universal (existential) quantifiers into CNFs (DNFs) *regardless.* However, this needlessly expands the scopes of the quantifiers. Instead, one should first apply PN laws to drive the quantifiers inwards as far as possible without converting their scopes into CNFs/DNFs. Only if the stated conditions are satisfied must the scopes be converted. Afterwards, the miniscoping process described above is applied again until the scopes of the quantifiers are no further reduced by this process.

Applying first PN9 and then PN3 to (7) results in

$$\exists y (\forall x (\neg Fxx \vee Hxy) \wedge (Gy \vee \forall x Hxy)) \wedge P \tag{8}$$

Reiterating once more the entire process described thus far requires applying DIS1 to convert the scope of $\exists y$ into a DNF. Applying PN10 subsequent to DIS1 finally results in the following disjunction of conjunctions of primary formulas:

$$\begin{gathered} \exists y (\forall x (\neg Fxx \vee Hxy) \wedge Gy) \wedge P \vee \\ \exists y (\forall x Hxy \wedge \forall x (\neg Fxx \vee Hxy)) \wedge P \end{gathered} \tag{9}$$

[6]This conversion is not considered in the miniscoping process described by Nonnengart & Weidenbach [11, p. 343f.]. This is why their miniscoping process (apart from ignoring quantifier sorting) does not result in formulas with scopes that are minimized to the maximal extent.

After the purification process is complete, the resulting expression is converted into an FOLDNF by applying DIS1. In the case of (9), no further application of DIS1 is necessary. The entire process of purification as described thus far may result in the replication of quantifiers. The replicated quantifiers bind variables of the same type. In the final step of the algorithm, the variables are renamed such that the variables bound by universal quantifiers in a given disjunct of the FOLDNF, in which m universal quantifiers occur, are denoted by $x_1 \dots x_m$. Likewise, the variables bound by existential quantifiers are denoted by $y_1 \dots y_n$, where n is the number of existential quantifiers that occur in the disjunct. Thus, no variable of a disjunct is bound by more than one quantifier in that disjunct (*maximal renaming*). For our purposes, it is sufficient to use different variables for different quantifiers within any disjunct of a FOLDNF. In the case of (9), maximal renaming results in the following FOLDNF:

$$\begin{gathered} \exists y_1(\forall x_1(\neg Fx_1x_1 \vee Hx_1y_1) \wedge Gy_1) \wedge P \vee \\ \exists y_1(\forall x_1 Hx_1y_1 \wedge \forall x_2(\neg Fx_2x_2 \vee Hx_2y_1)) \wedge P \end{gathered} \tag{10}$$

When the minimization of an FOLDNF is considered (cf. section 4.2), the variables are renamed such that the *minimum* number of different variables are used to identify the internal relations between primary formulas in any case (*minimal renaming*).

Thus, it is possible to implement a procedure to convert any formula ϕ of pure FOL into a formula relating completely purified formulas using the following algorithm:[7]

1. Convert ϕ into an NNF.
2. Apply the 1st miniscoping procedure.
3. Sort the quantifiers.
4. Sort the scopes.
5. Apply the 2nd miniscoping procedure.
6. Convert the scopes.
7. Reiterate steps 2-6 until no further changes occur.
8. Convert the resulting formula into an FOLDNF.
9. Rename the variables of each disjunct in accordance with the principle of maximal renaming.

The resulting FOLDNF is equivalent to the initial formula ϕ, as only equivalence rules are applied. The procedure terminates, as it is well known that conversion into NNFs and DNFs as well as the processes of renaming and sorting terminate. Furthermore, the miniscoping processes terminate, as they reduce the depth of a formula starting with a quantifier. The entire process results in disjunctions of conjunctions of primary formulas as defined above,

[7]For simplicity, I abstain from a more technical definition of the algorithm and instead refer simply to the names of the steps of the procedure as introduced in italics above. Aside from the equivalence rules used in the renaming process (cf. Nonnengart & Weidenbach [11, p. 344f.]), all of the logical equivalence rules involved are mentioned above, and the way in which they are applied in each step is trivial.

as PN1 to PN10 are applied to drive quantifiers inwards to the maximal extent, which is made possible by sorting strategies and scope conversions. The only factors limiting the driving in of quantifiers without applying equivalence laws to minimize formulas are the lack of equivalence rules for the cases of $\forall\nu(A(\nu) \vee B(\nu))$ and $\exists\mu(A(\mu) \wedge B(\mu))$ and the non-equivalence of changing the quantifier order in the case of $\exists\mu\forall\nu$ or $\forall\nu\exists\mu$.

Compared with a procedure that yields Hintikka's distributive normal forms[8], the described algorithm yields far less complex FOLDNFs and is, in fact, quite trivial and effective. The purification procedure (steps 1 to 7) is identical to Quine's procedure as presented in Quine [15, pp. 126-129] except for steps 3 to 6, which are omitted by Quine. Nonnengart & Weidenbach [11, pp. 341-343] refer only to steps 1 and 2. Hilbert & Bernays [4, p. 145] suggest to begin with PNFs and to minimize each quantifier sequentially from the inside to the outside of the prenex, starting with scope conversion (step 6) followed by miniscoping (step 2). However, this procedure satisfies neither condition (i) nor condition (ii) as stated above (cf. p. 5). It expands the scopes to the maximal extent and does not consider optimized quantifier orders. By contrast, the algorithm described in this section performs iterative optimized applications of PN laws to minimize the scopes to the maximal extent with the minimum number of necessary scope expansions. Because Hilbert & Bernays [4, pp. 144-148] as well as Quine [15, pp. 126-129] omit steps 3 to 6, they do not satisfy the condition of Definition 1 that *any* disjunction (conjunction) occurring in a primary formula must be preceded by universal (existential) quantifiers, all of which bind variables that occur in each disjunct (conjunct). This may be the reason why Hilbert & Bernays [4, p. 144] explicitly abstain from considering primary formulas of non-monadic FOL, as they judge such formulas to lack a sufficient degree of standardization.

The resulting FOLDNF obtained by applying steps 1 to 9 is a disjunction of conjunctions of purified (or primary) formulas satisfying Definition 2. The primary formulas of FOLDNFs are the equivalent of the literals in the DNFs of propositional logic. In the following, I first consider the purpose of such FOLDNFs in general (section 3) and then propose a procedure for minimizing them (section 4).

3 Explaining Truth Conditions

The outward form of arbitrary FOL formulas does not make it possible to identify or read off the truth conditions of their instances. This becomes clear when one considers their literal paraphrases: In most cases, such a paraphrase does not help to understand truth conditions. This situation changes immediately when one considers DNFs. Each disjunct of a DNF identifies a sufficient condition for truth within a logical space of possible truth conditions; the truth of at least one disjunct is a necessary condition for the truth of an instance of the initial formula. Each disjunct, in turn, enumerates the conditions for truth that must collectively be satisfied for the truth of the disjunct. In paraphrasing DNFs, it is not necessary to paraphrase logical constants literally. One can think of a disjunction as an enumeration of sufficient conditions for the truth of an instance of the initial formula and of a conjunction as an enumeration of necessary conditions for the truth of an instance of

[8]Hintikka was satisfied simply to have proven the existence of his distributive normal forms. Nelte [10, p. 81f.], however, describes an algorithm for converting an arbitrary formula ϕ into its Hintikka distributive normal form.

a disjunct. The negation sign occurs only to the left of atomic expressions (propositional functions) and may be paraphrased as indicating that certain atomic conditions are false (or not satisfied). Likewise, a lack of negation of certain atomic expressions indicates that the corresponding atomic conditions are true (satisfied). In this respect, a paraphrase of DNFs in the form of paraphrasing truth conditions serves as an analysans or explanans of an instance of a logical formula, this instance being the analysandum or explanandum. Consequently, the process of transforming formulas into FOLDNFs is a mechanical equivalence transformation within a process of analysis or explanation. The explanatory power of such an explanation arises from the resulting form of the explanans, which allows one to identify truth conditions.

Although the pure form of FOLDNFs already contributes to the identification of truth conditions, several questions and problems remain. Let us, for now, put aside intricate questions concerning internal relations between components of FOLDNFs that induce redundancies and motivate minimization. I address these questions in section 4. For simplicity, let us also, for now, exclude internal relations within disjuncts by allowing each predicate to occur only once in a given disjunct of an FOLDNF. In this case, only internal relations between disjuncts may arise, which we also will not address for the time being. We still need to clarify how primary formulas contribute to the identification of conditions for truth. Let us do so by considering the following example.

Let our initial formula ϕ be formula (1), p. 5:

$$\exists y \forall x(\neg(((\neg Fxx \land Gy) \lor Hxy) \to \neg P)) \quad (1)$$

It is obvious that a literal paraphrase that verbalizes the logical constants and the way in which they are connected is not of much help for understanding the truth conditions of this formula.

ϕ is converted into the FOLDNF given in formula (9), p. 7, according to the algorithm described in section 2. Formula (9), in turn, can be minimized to the following equivalent FOLDNF, composed of the two disjuncts (11) and (12):

$$\exists y_1 \forall x_1 Hx_1y_1 \land P \lor \tag{11}$$
$$\exists y_1(\forall x_1(\neg Fx_1x_1 \lor Hx_1y_1) \land Gy_1) \land P \tag{12}$$

Given this syntax, conditions for truth can be directly read off in a standardized way. Before explaining the paraphrase of (11) $\lor$ (12) in more detail, I note that from here on, I will abstain from cumbersome references to "instances" of formulas and simply speak of "truth conditions of formulas", tacitly meaning the truth conditions of their instances. As we are concerned with formulas and their contribution to the identification of truth conditions, this simplified terminology suffices for our purposes. Thus, we obtain the following paraphrase of (11) $\lor$ (12):

ϕ is true iff

– - Some object in the second position of Hx_1y_1 combined with all objects in the first position of Hx_1y_1 makes the dyadic propositional function Hx_1y_1 true, and
  - the atomic proposition P is true, or

-
  - Some object, the same in the second position of Hx_1y_1 and in the first position of Gy_1, combined with all objects distributed among (i) the first and second positions of Fx_1x_1 (where the same object satisfies these two positions) and (ii) the first position of Hx_1y_1, makes the dyadic propositional function Fx_1x_1 false, the dyadic propositional function Hx_1y_1 true, and the monadic propositional function Gy_1 true, and
  - the atomic proposition P is true.

In general, an existential quantifier within a primary formula indicates that one and the same object satisfies all positions of the propositional functions in which the variable that it binds occurs. In the case of universal quantifiers, one must distinguish whether their positions occur in propositional functions that are related by disjunctions or not. In the first case, a universal quantifier indicates that all objects are distributed among the corresponding positions; in the second case, a universal quantifier indicates that all of a set of distributed objects satisfy the corresponding positions. Furthermore, sequences of quantifiers refer to specific combinations of objects that satisfy certain conditions. Within a sequence of quantifiers, the order of the existential and universal quantifiers is significant. The order of existential (universal) quantifiers within sequences of existential (universal) quantifiers, however, is not significant. The same applies to quantifiers that occur in different sequences, as the quantifiers of one sequence are not within the scope of a quantifier of another. All these rules for interpreting primary formulas presume their standardized syntax. Unless the syntax is standardized in this way, it is not possible to establish unambiguous rules that allow one to read off truth conditions from the syntactic features of FOLDNFs.

Paraphrasing FOLDNFs in this way makes it evident that such paraphrases do not merely verbalize a logical notation but rather explain the truth conditions of logical formulas. This can be made even more evident by introducing an "iconic notation" for FOLDNFs that does not make use of the usual logical constants.[9] In this notation, the positions of variables are explicitly indicated by numbers, and bound variables are replaced with "forks" connecting numbers, thus explicitly symbolizing the relations of the bound variables to their positions (cf. table 5). By using an "open fork" (i.e., $\begin{smallmatrix}/\\ \\ \backslash\end{smallmatrix}$) to connect the positions of a universal variable occurring in different positions of disjunctive connected propositional functions, it is symbolized that the objects of a domain are distributed among those positions. By contrast, a closed fork (i.e., $<$) indicates that the same object must satisfy all positions that are connected by the fork. Furthermore, `T` or `F` indicates that a certain combination of objects makes some propositional function true or false (satisfies that function or not), respectively. Finally, disjuncts (conjunctions of length ≥ 1) are expressed as finite sets enumerating the iconic expressions of primary formulas, and a disjunction (of length ≥ 1) is represented by a finite set containing the enumeration of those sets. To such a set representing a disjunction, an outward `T` is assigned to indicate a set of conditions for truth.[10] These conventions make superfluous the use of the logical constants $\neg, \wedge$ and $\vee$ as well as the use of variables; they

[9] Both the idea of an iconic notation for FOL and some of the features of this iconic notation as summarized above can be traced back to Peirce's graphs (cf. Shin [18]), Wittgenstein's *ab*-notation (cf. Wittgenstein [20, p. 95f.]), and Quine's quantificational diagrams using bonds (cf. Quine [16, p. 70]).

[10] One can easily imagine a complementary set consisting of the conditions for the falsehood of ϕ. Such a complementary set can be generated either from the truth conditions of $\neg\phi$ or directly from the truth conditions of ϕ by applying rules for inversion.

explain the meaning of the logical constants by reducing the initial formula to its iconic symbol.

By applying the stated conventions, the FOLDNF (11) $\vee$ (12) can be translated into its iconic symbols, from inside to outside, via the following steps:

1. Translate the propositional functions (indicate positions with numbers that replace variables, and denote affirmation by `T` and negation by `F`): $Hxy \Rightarrow$ `T`-H_{12}, $P \Rightarrow$ `T`-P, $\neg Fxx \Rightarrow$ `F`-F_{12}, and $Gy \Rightarrow$ `T`-G_1.

2. Symbolize the relations of the bound variables to their positions. Use forks to connect the numbers of the positions as follows:

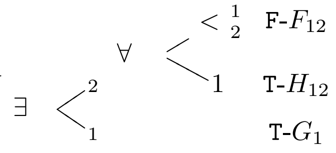
   (a) Open forks connect the numbers of positions connected by a disjunction and bound by a universal quantifier,

   (b) Closed forks connect the numbers corresponding to all other positions of one and the same variable.

   Thus, $\vee$ and $\wedge$ as well as the bound variables are eliminated in favour of forks connecting the positions of the corresponding propositional functions. In the case that a bound variable occurs only once within a propositional function, no fork is needed. This results in the translation of the primary formulas containing quantifiers:

   $$\exists y_1 \forall x_1 H x_1 y_1 \Rightarrow \exists_2 \forall_1 \ \texttt{T-}H_{12}$$

   $$\exists y_1(\forall x_1(\neg F x_1 x_1 \vee H x_1 y_1) \wedge G y_1) \Rightarrow$$

3. Translate the conjunctions of the primary formulas as sets of the translations of the primary formulas, and translate the disjunction of the conjunctions of the primary formulas as the set of the translations of the conjunctions. Prepend to the latter an outward `T` to indicate a set of truth conditions. This results in the iconic symbol of (11) $\vee$ (12), i.e., the iconic symbol for the following formula:

   $$\exists y_1 \forall x_1 H x_1 y_1 \wedge P \ \vee \exists y_1(\forall x_1(\neg F x_1 x_1 \vee H x_1 y_1) \wedge G y_1) \wedge P$$

   cf. table 5.

I abstain here from elaborating (i) the rules for translating traditional FOLDNFs into iconic notation and (ii) the rules for paraphrasing FOLDNFs in traditional or iconic notation in more detail. It is hoped that the iconic notation should speak for itself. Once one is accustomed to it, there is no need for the cumbersome and less transparent ordinary paraphrases of FOLDNFs. In contrast to ordinary FOLDNFs, the iconic notation replaces the ordinary notation of FOL with a notation that allows truth conditions to be identified from syntactic properties to the greatest possible extent. This is why the iconic notation provides an explanans for the truth conditions of the initial formula (the explanandum). Because it separates the sufficient and collectively necessary conditions, these conditions

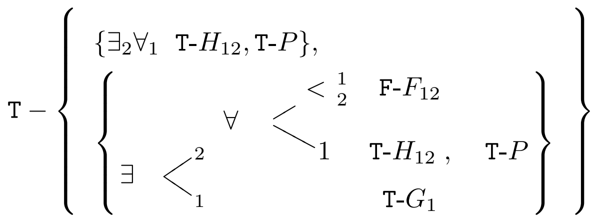


Table 5: Iconic symbol for (11) $\vee$ (12)

being composed of standardized primary expressions, it also provides an analysans of the initial formula (the analysandum).

The iconic expressions of disjuncts that *do not contain any propositional function more than once* can be regarded as providing syntactic criteria for identifying models given a suitable finite domain (an upper bound on the cardinality of such a domain is the maximal number of nested quantifiers that may appear within the primary formulas of the corresponding disjunct). In the case of the second set appearing in the iconic symbol presented in table 5, one must bear in mind that models depend on pairs that are *not* included in $\Im(F)$. To make this transparent, one can explicitly enumerate tuples that do not satisfy the corresponding propositional functions, where such tuples are indicated by an overline. To come to understand how models are identified by iconic symbols, one must focus on such overlined tuples in the case of propositional functions preceded by "F". Given such a representation, one can, for example, easily recognize how the following model, with a domain of two objects, $\{1, 2\}$, satisfies the syntactic criteria specified by the second set in the iconic symbol shown in table 5 by focusing on the underlined tuples:

$\Im(P)$**:** $\underline{\mathtt{T}}$,

$\Im(F)$**:** $\{\underline{\overline{(1,1)}}, (2,2), (1,2), (2,1)\}$,

$\Im(G)$**:** $\{\underline{1}, \overline{2}\}$,

$\Im(H)$**:** $\{\overline{(1,1)}, \overline{(1,2)}, \underline{\overline{(2,1)}}, \overline{(2,2)}\}$.

Such a model is more specific than the iconic symbol, as one must arbitrarily choose *specific* objects to satisfy the positions of the propositional functions. In this respect, models extend beyond the language of pure FOL and demand more than is necessary to identify truth conditions. They specify instances that make propositions true rather than the structural features that instances must satisfy for a proposition to be true.

It is possible to define a mechanical procedure for generating finite models by satisfying the conditions coded by the iconic notation for primary formulas of a disjunct that does not contain any propositional function more than once.[11] However, as this method in general applies only to disjuncts without internally related components and as the following discussion primarily concerns the minimization of FOLDNFs, I abstain from providing the

[11]In fact, I have implemented an algorithm to generate such models from FOLDNFs, in addition to the algorithm for generating minimized FOLDNFs described in section 4.3.

details here. For now, it is sufficient to mention this use of the logical notation for FOLDNFs to demonstrate the capability of FOLDNFs, and the iconic notation in particular, to enable the identification of truth conditions based on syntactic criteria. Contrary to a model-theoretic evaluation of a formula, the FOLDNF (11) $\vee$ (12) as well as the resulting iconic symbol can be paraphrased from outside to inside, thus identifying the properties that any model of the initial formula must satisfy.

These general remarks must suffice to explain how FOLDNFs express the truth conditions of formulas. There is no comparable means of explaining formulas in terms of enumerating the sufficient and collectively necessary conditions for truth. Model theory can do no more than provide instances of sufficient conditions for truth (or falsehood, in the case of counter-models). It is not based on an equivalence procedure within the realm of logic alone and thus goes beyond pure FOL. Compared with models, FOLDNFs of pure FOL provide only *partial descriptions*, in the sense that they do not mention specific objects. The representation of truth conditions (or, roughly speaking, of "possible worlds" or "knowledge") does not extend beyond the expressive power of the language used for such representation. In particular, it follows that there is no need to refer to infinite domains when representing truth conditions.[12] However, FOLDNFs provide not only a finite but also a neat way to code truth conditions (or represent knowledge). This is particularly true in the case of *minimized* FOLDNFs (cf. section 4).

Compared with Hintikka's distributive normal forms, the FOLDNFs generated by the algorithm described in section 2 are also partial descriptions in another sense: They do not enumerate exhaustive and mutually exclusive sufficient conditions for truth. Instead, they usually provide the sufficient and collectively necessary conditions for truth in a dense manner by ignoring the requirement for mutual exclusivity. This may have disadvantages for certain purposes because not all conditions for truth that can be separated are, in fact, explicitly separated. Furthermore, FOLDNFs, even completely minimized ones, do not satisfy the demand for an unambiguous representation in the sense of providing one and

[12]At least, this is true as long as one does not claim more than a correct representation of truth conditions. If one additionally claims the ability to *generate* models from disjuncts of FOLDNFs, it is only true as long as one does not consider disjuncts with propositional functions that occur more than once. This is because of the existence of disjuncts of FOLDNFs that are satisfied only within an infinite domain, e.g., the transformation of the axiomatic system described by Hilbert & Bernays [4, p. 14] into a conjunction of primary formulas:

$$\forall x_1 \exists y_1 F x_1 y_1 \wedge \forall x_2 \forall x_3 (\neg F x_2 x_3 \vee \forall x_4 (\neg F x_3 x_4 \vee F x_2 x_4)) \wedge \forall x_5 \neg F x_5 x_5 \tag{13}$$

If one further claims the ability to identify internal relations and logical properties from FOLDNFs, one must claim certain degrees of optimization to dispense with model theory and infinite models. Like model theory in the case of infinite domains, the optimization of FOLDNFs faces difficulties related to algorithmic realization; cf. section 4.1. However, questions concerning the optimization of FOLDNFs still refer not to infinite domains but rather to the optimization of finite expressions.

the same FOLDNF for all formulas of a class of equivalent formulas.[13] Satisfying such a claim within FOL would imply the need for a decision procedure. However, FOLDNFs may serve various purposes. In addition to the claims of (i) a completely specific representation or (ii) an unambiguous representation, one might also claim that the minimized FOLDNFs provide the neatest possible coding of truth conditions. Such a claim is very natural, not only for reasons of economy. It also respects the intuition that a representation of truth *conditions* should not contain any *irrelevant* features. Thus, all redundancies should be eliminated. From this consideration, the necessity of an algorithm for minimizing FOLDNFs arises. Such an algorithm becomes of prominent interest as soon as internal relations among the components of FOLDNFs are considered, as we do in the next section by discussing minimization strategies for FOLDNFs.

4 Minimization

In propositional logic, an algorithm for minimizing DNFs is readily available: the Quine-McCluskey algorithm. This algorithm is of significant importance in, e.g., engineering, information theory, and causal theory. Thus, a question arises regarding the extent to which it is possible to generalize such an algorithm. In the following, I first identify the problems with extending the Quine-McCluskey algorithm to the minimization of FOLDNFs and then describe a pragmatic algorithm of my own for minimizing (or optimizing) FOLDNFs.

4.1 Problems with Complete Minimization

For two reasons, a complete minimization algorithm for FOLDNFs is unrealistic: From a theoretical point of view, such an algorithm must presume the decidability of FOL, e.g., to eliminate inconsistent disjuncts. From a practical point of view, such an algorithm would have to be very complex. In propositional logic, all primary formulas (= atomic propositions and their negations) are consistent and non-tautologous, and they are logically independent of each other apart from their negations. In the general case of FOLDNFs, however, the complexity of minimization increases because the primary formulas of FOLDNFs may involve all kinds of logical dependencies:

- primary formulas may be tautologous or inconsistent (atomic inconsistency);
- a conjunction of consistent primary formulas, none of which is the negation of another, may still be inconsistent (implicit inconsistency);

[13] Within propositional logic, this demand is satisfied by the so-called reductive DNFs (RDNFs) that are obtained in the first step of the Quine-McCluskey algorithm starting from CDNFs. The complete minimization of DNFs of propositional logic, however, is an ambiguous problem in general. This can be demonstrated by considering the following example of propositional logic:

$$P \land \neg Q \lor \neg P \land Q \lor P \land R \lor Q \land R \tag{14}$$

$$P \land \neg Q \lor \neg P \land Q \lor P \land R \tag{15}$$

$$P \land \neg Q \lor \neg P \land Q \lor Q \land R \tag{16}$$

(15) and (16) are two equivalent completely minimized DNFs of the RDNF given in (14); cf. Quine [14]. Cf. also Kim [9, p. 67], who discusses the problems with identifying causally relevant factors based on minimized DNFs.

– one single primary formula may imply another single primary formula or may be subcontrary or contrary to another single primary formula, and moreover, the same applies to truth functions of primary formulas (diversity of logical relations);

– one conjunction/disjunction of primary formulas may imply another conjunction/disjunction of primary formulas, where all of the individual primary formulas are logically independent of each other (non-reducibility of logical relations to logical relations between single primary formulas); and

– not only may conjunctions and disjunctions of FOLDNFs be equivalent to proper components thereof, but also

  - primary formulas may be equivalent to primary formulas with fewer conjuncts or disjuncts (atomic minimization), and
  - a conjunction/disjunction of primary formulas may be equivalent to a conjunction / disjunction of minimized primary formulas (context sensitivity of minimization).[14]

Thus, in the context of FOLDNFs, problems arise that are not considered in the Quine-McCluskey algorithm for minimizing propositional DNFs, such as the identification of inconsistent primary formulas or inconsistent disjuncts and the minimization of primary formulas, disjuncts or disjunctions, because of relations of implication between primary formulas.

Furthermore, extending the two steps of the Quine-McCluskey algorithm to FOLDNFs would require more general and far more complex strategies. The first step of minimizing CDNFs of propositional logic in the Quine-McCluskey algorithm consists of merging disjuncts following the merging rule $A \wedge B \vee A \wedge \neg B \dashv\vdash A$ (where B is atomic and A is a conjunction of length ≥ 1). In FOL, the merging of disjuncts is possible via a more general rule: Given that a conjunction A_1 implies a conjunction A_2 and that the primary formulas B_1 and B_2 are subcontrary, then $A_1 \wedge B_1 \vee A_2 \wedge B_2 \dashv\vdash A_1 \vee A_2 \wedge B_2$. Thus, eliminating primary formulas within disjuncts of FOLDNFs based on internal relations with other disjuncts is a more complex task than the corresponding task in propositional logic. The same applies to eliminating components of a primary formula following some merging process (i.e., a process for minimizing disjuncts based on their relations with other disjuncts). Similar considerations apply to the second step of the Quine-McCluskey algorithm, which minimizes disjunctions (instead of disjuncts, as in step 1). Minimizing a disjunction within an FOLDNF based on its equivalence to a proper component of itself would require far more complex strategies than those adopted in the second step of the Quine-McCluskey algorithm.

Finally, a calculus for identifying internal relations (particularly relations of implication) within FOLDNFs is more complex than in the case of propositional logic because the relevant logical implications also rest on special relations between quantifiers and their relations to variables they bind. In particular, it is impossible to identify logical relations based on a complete calculus without involving at least one rule that increases the complexity of the formulas in question; cf. p. 19 below.

[14]For example:

$$\exists y_1(Fy_1 \wedge \forall x_1(Gx_1 \vee Hx_1y_1)) \wedge \forall x_1Gx_1 \dashv\vdash \exists y_1Fy_1 \wedge \forall x_1Gx_1 \tag{17}$$

Compared with the case of general FOLDNFs, minimizing DNFs in propositional logic is a very special case and a trivial process. However, the fact that both theoretical and practical arguments suggest that it is impossible to implement a general algorithm for minimizing FOLDNFs to the maximal extent does not mean that one should not seek a simplified algorithm that would allow one to minimize arbitrary FOLDNFs to *some* extent. On the one hand, we must admit that we do not have a full understanding of the truth conditions of a formula ϕ unless we are able to provide a completely minimized FOLDNF of ϕ. On the other hand, converting formulas into FOLDNFs and purging their (still unnecessarily intricate) representation of truth conditions of superfluous redundancies enhances the understanding of logical formulas, the more so as it enables the identification of the features that are relevant for specifying purified truth conditions. Thus, if we wish to explain logical formulas and what they contribute to the representation of truth conditions, then there is no other way but to provide minimized FOLDNFs, and the best we can do is to develop an algorithm that helps to do so in a reasonably pragmatic way.

In the following, I informally outline an algorithm that I have implemented to minimize FOLDNFs. On the one hand, this algorithm should be helpful in cases in which our understanding of FOL formulas is truly in need of help. This is why I do not confine the algorithm to fragments of FOL for which a complete minimization, or at least decidability, is available, such as monadic FOL or formulas that do not contain quantifiers with scopes containing dyadic sentential connectives. Instead, given a complexity with which even trained logicians are unable to easily cope, the algorithm should provide a minimized FOLDNF for an arbitrary formula ϕ of pure FOL. On the other hand, the algorithm should be suitably effective and provide outputs for formulas of some complexity in a reasonable amount of time. Admittedly, when the objective is conversion into FOLDNFs and the subsequent minimization of the result, initial formulas of great length (very roughly speaking, formulas of more than approximately 5 lines) that contain many internal relations (roughly speaking, formulas containing many occurrences of the same propositional function) are beyond what a sufficiently powerful algorithm can process in a reasonable amount of time. However, supposing that the initial formula is too complex to be grasped by a trained human but of sufficiently low complexity to be manageable by a computer, then the algorithm should provide a minimized FOLDNF within a reasonable amount of time (say, in most cases, within less than 10 minutes on a normal machine).

To this end, the algorithm I suggest restricts minimization according to the following rules:

1. It restricts minimization to

   (a) minimization of single primary formulas,

   (b) minimization of conjunctions of primary formulas based on internal relations between pairs of primary formulas within a given conjunction, and

   (c) minimization of disjunctions of conjunctions of primary formulas based on internal relations between primary formulas of pairs of disjuncts within a given disjunction.

   In particular, the algorithm does not provide any equivalent of the two steps of the Quine-McCluskey algorithm; it abstains from defining any equivalent for the intricate process of merging as well as from adopting strategies for identifying, in general, the equivalence of conjunctions or disjunctions to any of their proper components.

2. It makes use of a restricted, incomplete calculus to identify relations of implication between primary formulas.
3. It makes use of time constraint commands.

Item 1 above is explained in section 4.3; items 2 and 3 are explained in the following section.

4.2 Derivation Trees

Given the syntax of primary formulas, a correct and complete calculus identifying relations of implication between primary formulas can be defined by considering the *minimal syntactic differences* of the significant syntactic features of FOLDNFs. Table 6 provides a brief overview of the 13 rules of the calculus that specify those minimal syntactic differences between FOLDNFs that are truth-preserving. In fact, these rules are correct for any NNFs. Each rule concerns only a minimal syntactic difference between NNFs. The rules apply regardless of where the syntactic feature occurs within the formula. "E" denotes "elimination". "I" stands for "introduction". "μ/ν" means that the variable μ is replaced with a variable ν, which is new if ν does not appear to the left of "$\vdash$". "$\mu, \mu/\nu$" means that some (not necessarily all) of the occurrences of μ are replaced with ν.

$\exists\forall Ex$:	$\exists\mu\forall\nu \vdash \forall\nu\exists\mu$		
$\forall E1/\exists I1$:	$\forall\mu A(\mu) \vdash \exists\nu A(\mu/\nu)$		
$\forall E2$:	$\forall\mu\forall\nu A(\mu,\nu) \vdash \forall\mu A(\mu,\nu/\mu)$	$\exists I2$:	$\exists\mu A(\mu) \vdash \exists\mu\exists\nu A(\mu,\mu/\nu)$
$\forall E3$:	$\exists\mu\forall\nu A(\mu,\nu) \vdash \exists\mu A(\nu/\mu)$	$\exists I3$:	$\forall\mu A(\mu) \vdash \forall\mu\exists\nu A(\mu,\mu/\nu)$
$\vee E$:	$A \vee A \vdash A$	$\vee I$:	$A \vdash A \vee B$
$\wedge E$:	$A \wedge B \vdash A$	$\wedge I$:	$A \vdash A \wedge A$
$\bot E$:	$A \vee \bot \vdash A$	$\top I$:	$A \vdash A \wedge \top$

Table 6: Rules of implication

In fact, the algorithm does not make use of $\vee I$, $\wedge I$ or $\top I$ at all, and it applies the remaining rules only to identify relations of implication between primary formulas. To do so, sequences of existential quantifiers are defined as "orderless", i.e., the rules apply to any existential quantifier in the sequence. Existential quantifiers that are separated only by disjunctions or conjunctions are treated as existential quantifiers of one and the same sequence. Thus, only existential quantifiers separated by universal quantifiers are considered to belong to different sequences. The same applies to sequences of universal quantifiers. Thus, strictly speaking, the rules are also applied to conjunctions or disjunctions of primary formulas if those primary formulas are preceded by quantifiers of the same sort. It is always possible to group such quantifiers into one sequence by means of PN laws. Conjunctions and disjunctions are similarly orderless, and $\wedge E$, $\vee E$ and $\bot E$ are also applied to conjunctions of primary formulas or to disjunctions of conjunctions of primary formulas. The resulting formula on the right-hand side of "$\vdash$" need not be a primary formula (or an FOLDNF). If it is not, it must be converted into a primary formula (or an FOLDNF) through equivalence transformation. This may result in conjunctions or disjunctions of primary formulas, to which the rules may, in turn, be applied. In this way, primary formulas that are implied by other primary formulas can be identified through iterative application of the rules via conjunctions/disjunctions of primary formulas.

The correctness of the calculus can be proven either by paraphrasing or by translation into well-known logical rules applying to NNFs. The completeness of the rules can be proven by considering all (finite) additional possible *minimal* syntactic differences between FOLDNFs and by demonstrating, either by paraphrasing or based on model theory, that they are not truth-preserving. However, as we do not claim that the minimization algorithm is complete, we need not go into detail here.

Instead, the algorithm makes use of derivation trees, starting from primary formulas and iteratively applying only 10 of the 13 rules of implication in all possible ways (cf. figure 1 for a rather simple example). The three rules $\vee I$, $\wedge I$ and $\top I$ all increase the complexity of the formula and are thus omitted to ensure the generation of derivation trees of finite size. In addition, a time constraint command is used to restrict the generation of the derivation trees in complex cases. The time required to minimize an FOLDNF essentially depends on the number of derivation trees that are generated, which, in turn, depends on the number of primary formulas of the FOLDNF. As this number is finite and the generated derivation trees are finite in size, the minimization procedure is guaranteed to terminate. Time constraint commands ensure that the procedure terminates in a reasonable amount of time given FOLDNFs of a reasonable complexity.

Whereas one can compensate for not using $\vee I$ by applying refined rules for identifying whether one primary formula is implied by another (cf. p. 22), refraining from using $\wedge I$ and $\top I$ makes it impossible to identify relations of implication between certain primary formulas. A single example may suffice to demonstrate this. The following formula is a conjunction of two primary formulas:

$$\begin{aligned}&\exists y_1(\neg Fy_1 \wedge \exists y_2(Fy_2 \wedge \exists y_3(\neg Gy_1y_3 \wedge \neg Gy_3y_2))) \wedge \\ &\qquad \forall x_1(Fx_1 \vee \forall x_2(\neg Fx_2 \vee Gx_1x_2)) \end{aligned} \tag{18}$$

This conjunction is inconsistent, as shown by the fact that the first primary formula implies the negation of the second (converted into a primary formula, i.e., the formula in line (5) of table 7). However, to prove this, one must apply either $\wedge I$ or $\top I$ in addition to some of the 10 rules used for the construction of derivation trees. In the derivation presented in table 7, $\top I$ is applied.

In addition, $\forall E3$ is applied to replace x_1 with y_3 in the formula appearing in line (2), which is possible because $\exists y_1 \exists y_2 \exists y_3$ is an orderless sequence by definition. The result of $\forall E3$ is transformed into a disjunction of primary formulas in line (3). $\wedge E$ is then applied to eliminate $\neg Fy_1$ and $\neg Gy_1y_3$ in the first disjunct of the formula in line (3) and to eliminate Fy_2 and $\neg Gy_3y_2$ in the second disjunct. Finally, $\vee E$ eliminates one of the two identical disjuncts in line (4). The variables are renamed to standardize the variable usage in each

$$\begin{tikzcd}
 & & \forall x_1 \forall x_2 F x_1 x_2 \arrow[dll, "\forall E2"'] \arrow[d, "\forall E1/\exists I1"] \arrow[drr, "\forall E1/\exists I1"] & & \\
\forall x_1 F x_1 x_1 \arrow[d, "\forall E1/\exists I1"'] \arrow[drr, "\exists I3"] \arrow[drrrr, "\exists I3"] & & \exists y_1 \forall x_1 F y_1 x_1 \arrow[dll, "\forall E3"'] \arrow[d, "\exists\forall E x"] & & \exists y_1 \forall x_1 F x_1 y_1 \arrow[dllll, "\forall E3"'] \arrow[d, "\exists\forall E x"] \\
\exists y_1 F y_1 y_1 \arrow[drr, "\exists I2"'] & & \forall x_1 \exists y_1 F y_1 x_1 \arrow[d, "\forall E I/\exists I1"] & & \forall x_1 \exists y_1 F x_1 y_1 \arrow[dll, "\forall E1/\exists I1"] \\
 & & \exists y_1 \exists y_2 F y_1 y_2 & &
\end{tikzcd}$$

Figure 1: Derivation tree starting from $\forall x_1 x_2 F x_1 x_2$

(1)	$\exists y_1(\neg Fy_1 \wedge \exists y_2(Fy_2 \wedge \exists y_3(\neg Gy_1y_3 \wedge \neg Gy_3y_2)))$	(18)
(2)	$\exists y_1(\neg Fy_1 \wedge \exists y_2(Fy_2 \wedge \exists y_3(\neg Gy_1y_3 \wedge \neg Gy_3y_2))) \wedge \forall x_1(Fx_1 \vee \neg Fx_1)$	$\top I$
(3)	$\exists y_1(\neg Fy_1 \wedge \exists y_2(Fy_2 \wedge \exists y_3(\neg Gy_1y_3 \wedge \neg Gy_3y_2 \wedge Fy_3))) \vee$ $\exists y_1(\neg Fy_1 \wedge \exists y_2(Fy_2 \wedge \exists y_3(\neg Gy_1y_3 \wedge \neg Gy_3y_2 \wedge \neg Fy_3)))$	$\forall E3$
(4)	$\exists y_1(\neg Fy_1 \wedge \exists y_2(Fy_2 \wedge \neg Gy_1y_2)) \vee$ $\exists y_1(\neg Fy_1 \wedge \exists y_2(Fy_2 \wedge \neg Gy_1y_2))$	$4 \times \wedge E$
(5)	$\exists y_1(\neg Fy_1 \wedge \exists y_2(Fy_2 \wedge \neg Gy_1y_2))$	$\vee E$

Table 7: Derivation using $\top I$

case.

Thus, abstaining from the application of $\wedge I$ and $\top I$ implies incompleteness. Consequently, using this approach prevents complete minimization. However, if one were to permit the iterative application of $\wedge I$ or $\top I$, then the size of the derivation trees would increase to infinity. By relying on the derivation trees generated by all possible applications of the remaining 10 of the 13 rules listed in table 6, the minimization strategy achieves termination at the expense of complete minimization.

4.3 FOL Optimizer

I call the program implemented to generate a minimized FOLDNF for a given formula ϕ of pure FOL "FOL Optimizer". Roughly speaking, the program proceeds as follows (cf. figure 2):

1. Transform ϕ into an FOLDNF using the algorithm described in section 2.
2. Minimize single primary formulas:
   (a) Eliminate redundant conjuncts within primary formulas.
   (b) Eliminate redundant disjuncts within primary formulas.
   (c) Eliminate an entire primary formula if its negation can be identified as inconsistent.[15]
3. Minimize single disjuncts of the result obtained in step 2:
   (a) Eliminate an entire disjunct if some primary formula thereof can be identified as inconsistent.
   (b) Eliminate an entire disjunct if some primary formula thereof implies the negation of some other primary formula thereof.[16]
   (c) Eliminate a single primary formula if it is implied by some other primary formula of the same disjunct.
4. Minimize the resulting disjunction obtained in step 3:
   (a) If two disjuncts of length 1 contain primary formulas that can be identified as subcontrary (i.e., the negation of primary formula A implies primary formula B), then return a result of "True" (indicating a tautology).
   (b) Eliminate a disjunct $D1$ if all primary formulas of another disjunct $D2$ are implied by the primary formulas of $D1$.

The algorithm returns a result of "False" in the case that ϕ can be identified as inconsistent and a result of "True" in the case that ϕ can be identified as tautologous; otherwise, it returns a minimized FOLDNF that is equivalent to the initial formula ϕ.

[15]If all primary formulas of a disjunct of the FOLDNF are thus identified as tautologous, then ϕ is identified as tautologous.

[16]If all disjuncts of the FOLDNF are identified as inconsistent, then ϕ is identified as inconsistent.

Step 1, in fact, also involves the application of well-known propositional minimization procedures (beyond the algorithm described in section 2). These minimization procedures and those applied in steps 2 to 4 all minimize conjuncts or disjuncts within FOLDNFs (and thus NNFs) and rely on the following equivalence rules (from left to right):

$A \wedge A$	$\dashv\vdash$	A		IP1a$\wedge$
$A \vee A$	$\dashv\vdash$	A		IP1a$\vee$
$(A \vee B) \wedge (A \vee \neg B)$	$\dashv\vdash$	A		IP2$\wedge$
$(A \wedge B) \vee (A \wedge \neg B)$	$\dashv\vdash$	A		IP2$\vee$
$\top \wedge A$	$\dashv\vdash$	A		IP1$\top$
$\top \vee A$	$\dashv\vdash$	$\top$		IP2$\top$
$\bot \wedge A$	$\dashv\vdash$	$\bot$		IP1$\bot$
$\bot \vee A$	$\dashv\vdash$	A		IP2$\bot$
$A \wedge B$	$\dashv\vdash$	A	if $A \vdash B$	IP1b$\wedge$
$A \vee B$	$\dashv\vdash$	A	if $B \vdash A$	IP1b$\vee$

Table 8: Equivalence rules used for minimization

The last two rules are used only in steps 2 to 4 of the algorithm. Whether the conditions $A \vdash B$ and $B \vdash A$ for these two rules hold is decided by referring to derivations trees; cf. section 4.2. Reference to derivation trees is also made in steps 2 to 4 to identify the conditions $\top$ and $\bot$ for IP1/2$\top$ and IP1/2$\bot$. To apply all stated minimization rules to the maximal extent, *minimal renaming* (cf. p. 8) is used.

In addition to the stated rules, we tacitly assume the following equivalence rules:

$A \vee \neg A$	$\dashv\vdash$	$\top$	R1
$A \wedge \neg A$	$\dashv\vdash$	$\bot$	R2
$\top$	$\dashv\vdash$	$\neg \bot$	R3
$A \wedge B$	$\dashv\vdash$	$\bot$ if $A \vdash \neg B$	R4
$A \vee B$	$\dashv\vdash$	$\top$ if $\neg A \vdash B$	R5

Table 9: Tacitly assumed equivalence rules

In step 2(a), redundant conjuncts within a primary formula A are identified by substituting single conjuncts of a conjunction C with other single conjuncts of C. Each such substitution results in a primary formula A'. For each such formula A', a derivation tree is generated. Furthermore, from the primary formula A, all possible formulas are generated such that disjuncts within A are deleted. The resulting set of formulas, including A, is denoted by A^*. The set A^* is generated because $\vee I$ is not applied for the generation of derivation trees. However, A' implies A if A' implies some formula in the set A^*, because all formulas in A^* imply A. As a substitute for deriving A from A' by using $\vee I$, the formulas in A^* can be derived from A' without using $\vee I$. This "trick" for circumventing the use of $\vee I$ is also used in all other steps of the algorithm when the objective is to test whether some primary formula is implied by another primary formula using derivation trees. Furthermore, the variables are always standardized by means of minimal renaming; cf. p. 8. This enables the algorithm to decide whether some primary formula X is implied by another

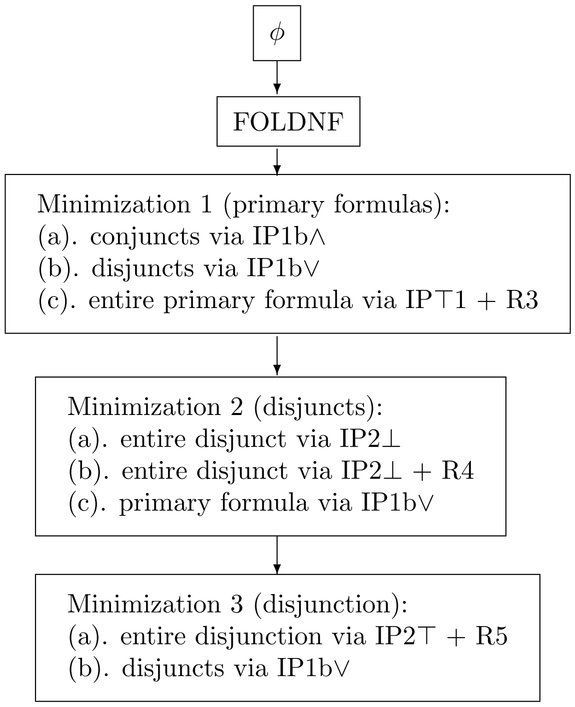

ϕ
FOLDNF
Minimization 1 (primary formulas):
(a). conjuncts via IP1b$\wedge$
(b). disjuncts via IP1b$\vee$
(c). entire primary formula via IP$\top$1 + R3
Minimization 2 (disjuncts):
(a). entire disjunct via IP2$\perp$
(b). entire disjunct via IP2$\perp$ + R4
(c). primary formula via IP1b$\vee$
Minimization 3 (disjunction):
(a). entire disjunction via IP2$\top$ + R5
(b). disjuncts via IP1b$\vee$

Figure 2: Chart of the algorithm

primary formula Y by virtue of the 10 rules applied to generate a derivation tree for Y by checking whether X is a member of that tree. Redundant conjuncts are thus identified by checking whether some (standardized) formula of A^* is a member of the derivation tree of A'. If a conjunct is redundant, it is eliminated; the resulting formula is transformed into a primary formula through equivalence transformation, and its variables are standardized by means of minimal renaming. This process is iteratively applied for all single conjuncts of all conjunctions of all primary formulas. To limit the construction of derivation trees, trees are generated only if the conjunct in question and the conjunct that is substituted for it contain at least one literal of the same length, involving the same predicate letter and preceded by a negation sign in either both cases or neither. This is a trivially necessary condition for the redundancy of a conjunct based on its internal relation to another conjunct in the same conjunction. Furthermore, the generation of derivation trees terminates as soon as some member of A^* is a member of the tree.[17]

Step 2(b) proceeds in a manner analogous to that of step 2(a). Because disjuncts instead of conjuncts are being considered, a derivation tree is generated for A, and the algorithm tests whether some formula of the set of formulas A'^* is a member of that tree.

[17]In addition, the generation of derivation trees is always restricted by a time constraint of 10 seconds, which is sufficient unless the primary formulas are quite complex. The generation of derivation trees also terminates if an explicit contradiction (identifiable based on propositional contradictions of quantifier scopes) is derived. In this case, the corresponding primary formula, and thus the disjunct that contains it, is identified as inconsistent. Consequently, the inconsistent disjunct is deleted unless it is the only disjunct of the FOLDNF, in which case the initial formula ϕ is identified as inconsistent.

In step 2(c), tautologous primary formulas are identified by generating the derivation trees for their negations converted into standardized primary formulas. If some explicit contradictory formula can be identified within a derivation tree, then the primary formula under consideration is tautologous. Several plain necessary conditions for inconsistency are implemented to limit the generation of derivation trees.

In step 3, derivation trees of primary formulas are generated from single disjuncts of the FOLDNF obtained in step 2. As soon as a derivation tree contains an explicit contradiction (step 3(a)) or the negation of some other primary formula, where this negation is standardized and converted into a primary formula (step 3(b)), the corresponding disjunct is identified as inconsistent and is deleted. Otherwise, a primary formula A is deleted if some formula of the set A^* is a member of some derivation tree of another primary formula of the same disjunct (step 3(c)).

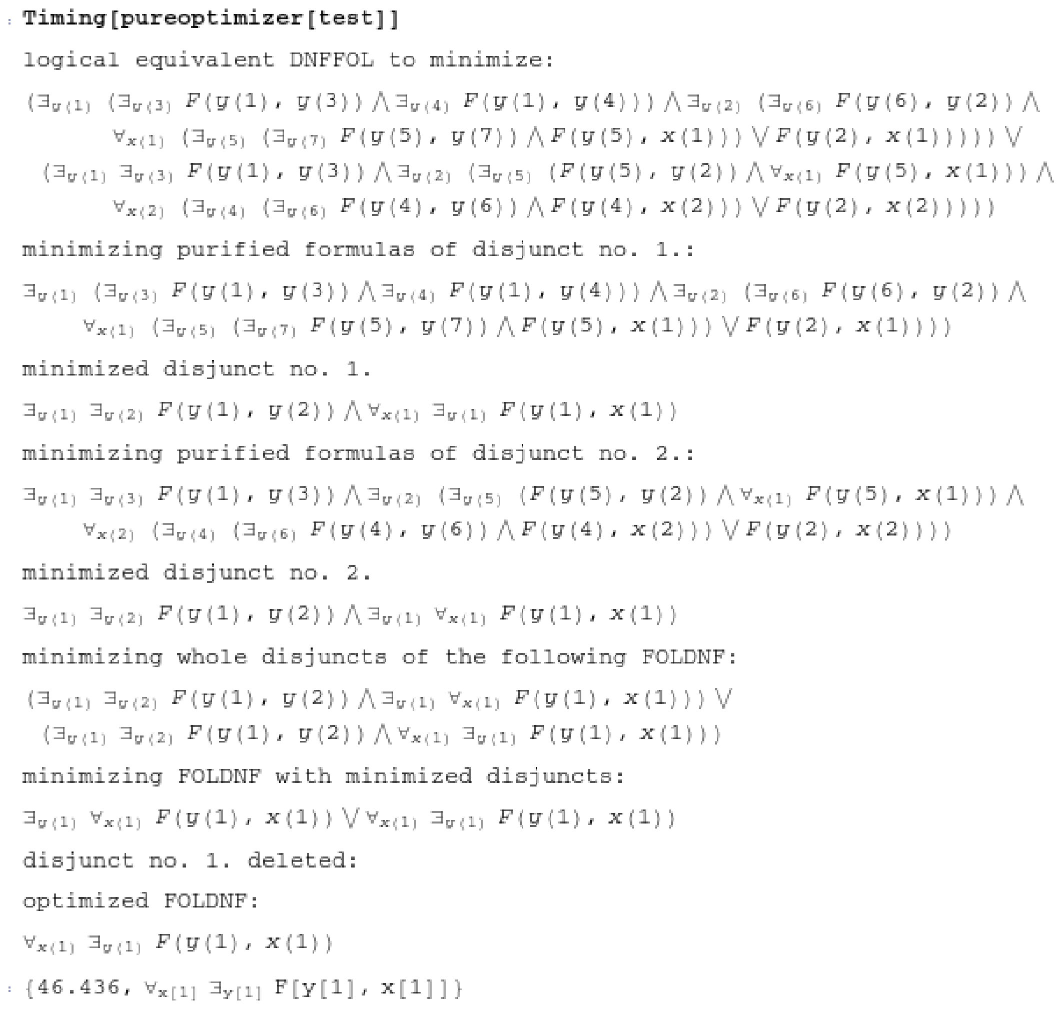
In step 4, derivation trees of primary formulas of the resulting FOLDNF obtained in step 3 are generated to identify internal relations between primary formulas of different disjuncts. Step 4(a) involves a simple test for tautologies with respect to relations of implication between single primary formulas of different disjuncts. If the derivation tree of the negation of a primary formula A (standardized and transformed into a primary formula) contains some formula in the set A'^* of a primary formula A', where A and A' are distinct disjuncts of the FOLDNF resulting from step 3, then the initial formula ϕ is a tautology. In step 4(b), a disjunct $D1$ is deleted if each set A'^* of primary formulas of another disjunct $D2$ contains some member that is a member of the union of the derivation trees of the primary formulas of $D1$. Again, several plain criteria are implemented to restrict the generation of derivation trees in step 4.

The algorithm can be trivially seen to terminate because (i) step 1 (transformation into an FOLDNF) terminates (cf. p. 8) and (ii) steps 2-4 successively minimize finite parts of the FOLDNF according to decisions regarding the implication relations between primary formulas, which terminate because of time constraint commands and the restriction to (incomplete) derivation trees of finite size (cf. p. 19). The algorithm is correct, as it involves the application of nothing but logical equivalence rules. Finally, it serves its expected purpose (namely, minimizes an FOLDNF to *some* extent), as it first transforms an initial formula ϕ into an FOLDNF and then uses logical equivalence rules to minimize that FOLDNF by eliminating conjuncts and/or disjuncts.

Figure 3 illustrates the results of the various steps of the algorithm for the following example `test`:

$$\neg\forall y_6\neg\neg\forall y_3\neg\exists y_2\exists y_4\forall x_1\neg(\neg Fy_3y_3 \lor \neg(\forall x_2(\exists y_1 Fy_2y_1 \land \\ (\neg Fy_2y_4 \to Fy_3x_1) \land \exists y_5\exists y_7((Fy_5y_7 \land Fy_5x_2) \lor Fy_6x_2)))) \quad \texttt{test}$$

The truth conditions of instances of `test` are difficult to grasp. A literal paraphrase of `test` provides no assistance. Transforming `test` into an FOLDNF according to step 1 of the FOL Optimizer procedure is of some help. However, the result contains many redundancies that still obscure the truth conditions. Subsequent stepwise minimization makes the truth conditions increasingly clear. The result, $\forall x_1\exists y_1 Fx_1y_1$ (or $\mathtt{T} - \{\{\forall_1\exists_1\mathtt{T} - F_{12}\}\}$ in iconic notation), finally explains the truth conditions for the initial formula `test`.



```
Timing[pureoptimizer[test]]
logical equivalent DNFFOL to minimize:
(∃y(1) (∃y(3) F(y(1), y(3)) ⋀ ∃y(4) F(y(1), y(4))) ⋀ ∃y(2) (∃y(6) F(y(6), y(2)) ⋀
      ∀x(1) (∃y(5) (∃y(7) F(y(5), y(7)) ⋀ F(y(5), x(1))) ⋁ F(y(2), x(1))))) ⋁
  (∃y(1) ∃y(3) F(y(1), y(3)) ⋀ ∃y(2) (∃y(5) (F(y(5), y(2)) ⋀ ∀x(1) F(y(5), x(1))) ⋀
      ∀x(2) (∃y(4) (∃y(6) F(y(4), y(6)) ⋀ F(y(4), x(2))) ⋁ F(y(2), x(2)))))
minimizing purified formulas of disjunct no. 1.:
∃y(1) (∃y(3) F(y(1), y(3)) ⋀ ∃y(4) F(y(1), y(4))) ⋀ ∃y(2) (∃y(6) F(y(6), y(2)) ⋀
    ∀x(1) (∃y(5) (∃y(7) F(y(5), y(7)) ⋀ F(y(5), x(1))) ⋁ F(y(2), x(1))))
minimized disjunct no. 1.
∃y(1) ∃y(2) F(y(1), y(2)) ⋀ ∀x(1) ∃y(1) F(y(1), x(1))
minimizing purified formulas of disjunct no. 2.:
∃y(1) ∃y(3) F(y(1), y(3)) ⋀ ∃y(2) (∃y(5) (F(y(5), y(2)) ⋀ ∀x(1) F(y(5), x(1))) ⋀
    ∀x(2) (∃y(4) (∃y(6) F(y(4), y(6)) ⋀ F(y(4), x(2))) ⋁ F(y(2), x(2))))
minimized disjunct no. 2.
∃y(1) ∃y(2) F(y(1), y(2)) ⋀ ∃y(1) ∀x(1) F(y(1), x(1))
minimizing whole disjuncts of the following FOLDNF:
(∃y(1) ∃y(2) F(y(1), y(2)) ⋀ ∃y(1) ∀x(1) F(y(1), x(1))) ⋁
  (∃y(1) ∃y(2) F(y(1), y(2)) ⋀ ∀x(1) ∃y(1) F(y(1), x(1)))
minimizing FOLDNF with minimized disjuncts:
∃y(1) ∀x(1) F(y(1), x(1)) ⋁ ∀x(1) ∃y(1) F(y(1), x(1))
disjunct no. 1. deleted:
optimized FOLDNF:
∀x(1) ∃y(1) F(y(1), x(1))
{46.436, ∀x[1] ∃y[1] F[y[1], x[1]]}
```

Figure 3: Optimization of `test` using FOL Optimizer

Acknowledgement

I am grateful to Markus Säbel for comments on a previous version of this paper. I would also like to thank the anonymous referees for their helpful suggestions.

References

[1] Behmann, H.: “Beiträge zur Algebra der Logik, insbesondere zum Entscheidungsproblem”, *Mathematische Annalen* 86, 1922, 163-229.

[2] Church, A.: "Review: G.H.v. Wright: On the idea of logical truth (I); G.H.v. Wright: Form and Content in Logic", *The Journal of Symbolic Logic* 15, 1950, 58-59.

[3] Ehrenfeucht, A.: "An application of games to the completeness problem for formalized theorems", *Fundamenta Mathematicae* 49, 1961, 129-141.

[4] Hilbert, D. & Bernays, P.: *Grundlagen der Mathematik I*, Springer, Berlin u.a., 1928.

[5] Hintikka, J.:"Distributive normal forms in the calculus of predicates", *Acta Philosophica Fennica* 6, 1953.

[6] Hintikka, J.: "Distributive normal forms in first-order logic", in: Crossley, J.N. & Dummett, M.A.E. (eds.), *Formal Systems and Recursive Functions*, North-Holland Publishing Company, Amsterdam, 1965, pp. 48-91.

[7] Hintikka, J.: "Are logical truths analytic?", *The Philosophical Review* 74(2), 1965, 178-203.

[8] Hintikka, J.: *Logic, Language-Games and Information*, Clarendon, Oxford, 1973.

[9] Kim, J.: "Causes and events: Mackie on causation", in: Sosa, E. & Tooley, M. (eds.), *Causation*, Oxford University Press, Oxford, 1993, pp. 60-74.

[10] Nelte, K.: Formulas of First-Order Logic in Distributive Normal Form, University of Cape Town, 1997: https://open.uct.ac.za/handle/11427/9648, 1997.

[11] Nonnengart, A. & Weidenbach, C.: "Computing small clausal normal forms", in: Robinson, A. & Voronkov, A. (eds.), *Handbook of Automated Reasoning I*, Elsevier, Amsterdam u.a., 2001, pp. 335-367.

[12] Oglesby, F.C.: *An Examination of a Decision Procedure*, Memoirs of the American Mathematical Society Number 44, American Mathematical Society, Providence, RI, 1962.

[13] Quine, W.V.O.: "On the logic of quantification", *Journal of Symbolic Logic* 10(1), 1945, 1-12.

[14] Quine, W.V.O.: "On core and prime implicants of truth functions", *American Mathematical Monthly* 66, 1959, 755-760.

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

[16] Quine, W.V.O.: *Mathematical Logic*, Revised Edition, Harvard University Press, Cambridge, MA, 1983.

[17] Scott, D.: "A note on distributive normal forms", in: Saarinen, E., Hilpinen, R., Niiniluotu, I. & Provence Hintikka, M. (eds.), *Essays in Honour of Jaakko Hintikka*, Reidel, Dordrecht, 1979, pp. 75-90.

[18] Shin, S.J.: *The Iconic Logic of Peirce's Graphs*, MIT Press, Cambridge, MA, 2002.

[19] Wittgenstein, L.: *Tractatus Logico-Philosophicus*, Routledge, London, 1974.

[20] Wittgenstein, L.: “Notes on logic”, in: Wittgenstein, L., *Notebooks 1914-19*, Blackwell, Oxford, 1979, pp. 93-107.

[21] Wright, G.H.v.: “On the idea of logical truth (I).” *Societas Scientiarum Fennica, Commentationes physico-mathematicae* 14(4), Helsingfors, 1948, 1-20, reprinted in Wright (1957).

[22] Wright, G.H.v.: *Form and Content in Logic*, Cambridge University Press, Cambridge, UK, 1949, pp. 1-35, reprinted in Wright (1957).

[23] Wright, G.H.v.: *Logical Studies*, Routledge, London, 1957.

[24] Wright, G. H. v.: *Wittgenstein*, Oxford, Blackwell, 1982.