On Calculational Proofs

Vladimir Lifschitz
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
vl@cs.utexas.edu

Abstract

This note is about the "calculational style" of presenting proofs introduced by Dijkstra and Scholten and adopted in some books on theoretical computer science. We define the concept of a calculation, which is a formal counterpart of the idea of a calculational proof. The definition is in terms of a new formalization **DS** of predicate logic. Any proof tree in the system **DS** can be represented as a sequence of calculations. This fact shows that any logically valid predicate formula has a calculational proof.

1 Introduction

This note is about the calculational proof format introduced by Dijkstra and Scholten [1990] on the basis of ideas due to W. H. J. Feijen. Their work has had an impact on the presentation of proofs in computer science research, and even on undergraduate education in computer science (as the result, in particular, of the publication of a textbook by Gries and Schneider [1993]). These ideas can be also viewed as an interesting discovery in logic, although this fact has so far remained largely unnoticed.

We know that inventing proofs of predicate formulas in a Hilbert style deduction system is an unpleasant task unless we use some metamathematical tools such as the deduction theorem. On the other hand, proving formulas in a natural deduction system is relatively easy. The role of the deduction theorem and of natural deduction is determined by the fact that they provide a mechanism for discharging assumptions.

What we learn from Dijkstra and Scholten is that the importance of assumptions in formal proofs is overrated: *with the right choice of inference* *rules, proving predicate formulas is easy even without introducing assumptions* (and without the use of auxiliary derivable objects such as sequents). According to our analysis of calculational proofs, these surprisingly powerful inference rules[1] are

$$\frac{F(G) \quad G \equiv H}{F(H)} \qquad \text{and} \qquad \frac{F(H) \quad G \equiv H}{F(G)}\,, \tag{1}$$
$$\text{where } G,\ H \text{ are free for a propositional symbol } P \text{ in } F(P).$$

In propositional logic, these are actually the only rules needed to derive arbitrary tautologies from a small set of axiom schemas.

Accumulating a supply of equivalences proved using rules (1) gives us the same feeling as accumulating a supply of identities in high school algebra or trigonometry: the more we have proved, the greater is the assortment of tools at our disposal when we approach a new problem. The more equivalences we have derived in predicate logic, the more choices we have for the second premise when rules (1) are used to prove new formulas.

Here is how Dijkstra and Scholten describe their experiences with this process:

> ... we approached the task of designing the theory we needed as an exercise in formal mathematics, little suspecting that we were heading for a few of the most pleasant surprises in our professional lives... [W]e found ourselves in possession of a tool that surpassed our wildest expectations. As we got used to it, it became an absolute delight to work with. The first pleasant... experience was the killing of the myth that formal proofs are of necessity long, tedious, laborious, error-prone, and what-have-you... For quite a while each new and surprisingly effective proof was a source of delight and excitement... In the course of the process we profoundly changed our ways of doing mathematics, of teaching it, and of teaching how to do it... [A] revision of mathematical methodology seemed at first a rather presumptious undertaking. As time went on, however, we were forced to conclude that the formal techniques we were trying out had never been given a fair chance...

(pp. v–vi).

[1] The use of these rules for formalizing calculational proofs was independently proposed by Anthony Hoare (David Gries, personal communication).

Calculational proofs are indeed a new thing in logic, just as natural deduction was a new thing when it was first introduced by Gentzen. One reason why this contribution has not been appreciated by logicians is that what Dijkstra and Scholten call formal proofs are not really formal proofs in the usual sense; they are informal proofs written in a relatively strict format.[2]

In this note, we define the concept of a "calculation," which is a formal counterpart of Dijkstra—Scholten calculational proofs. Calculations are defined in terms of a new formalization **DS** of predicate logic whose inference rules are replacement rules (1), the usual generalization rule

$$\frac{F}{\forall x F} \tag{2}$$

and the "particularization" rule

$$\frac{\forall x F(x)}{F(t)}\,, \quad \text{where } t \text{ is free for } x \text{ in } F(x). \tag{3}$$

The system **DS** is sound and complete, and any proof tree in **DS** can be represented as a sequence of calculations. These facts show that any logically valid predicate formula has a "calculational style proof."

2 The Deductive System

In the language of **DS**, *terms* and *atomic formulas* are defined as in predicate logic (without equality). *Formulas* are built from atomic formulas and the 0-place connective $\perp$ ("false") using the binary connectives $\equiv$ and $\vee$ and the quantifier $\forall$.

There are 9 axiom schemas in **DS**: equivalence is commutative

$$(F \equiv G) \equiv (G \equiv F) \tag{4}$$

and associative

$$(F \equiv (G \equiv H)) \equiv ((F \equiv G) \equiv H); \tag{5}$$

[2]Dijkstra and Scholten [1990] warn the readers that their book "is *not* a treatise on logic... [T]he logician is perfectly free to be taken aback by our naive refusal to make some of his cherished distinctions" (p. vii). One of the distinctions the authors ignore is the difference between formulas and inference rules, and it is impossible to define a formal system without it.

disjunction is commutative

$$F \vee G \equiv G \vee F, \tag{6}$$

associative

$$F \vee (G \vee H) \equiv (F \vee G) \vee H \tag{7}$$

and idempotent

$$F \vee F \equiv \; F; \tag{8}$$

it has a neutral element

$$F \vee \bot \equiv F, \tag{9}$$

distributes over equivalence

$$F \vee (G \equiv H) \equiv (F \vee G \equiv F \vee H), \tag{10}$$

and commutes with the universal quantifier

$$\forall x(F \vee G(x)) \equiv F \vee \forall x G(x), \text{ where } x \text{ is not free in } F. \tag{11}$$

The inference rules are (1), (2) and (3).

It is clear that this deductive system is sound. Its completeness is proved in Appendix.

As an example, note that the reflexivity of equivalence

$$F \equiv F \tag{12}$$

can be proved in **DS** in one step, using the first replacement rule:

$$\frac{F \vee \bot \equiv F \qquad F \vee \bot \equiv F}{F \equiv F}\,. \tag{13}$$

3 Calculations

A *calculation* is a derivation tree in **DS** such that every node in it belongs to the leftmost branch or is a leaf. The formulas in the leftmost branch are the *members* of the calculation; the formulas occurring in the other nodes of the derivation are the *hints*. Obviously, every hint is an equivalence. Two members of a calculation have special names: the one in the leftmost leaf is the *source* of the calculation, and the one in the root node is the *destination*. The hints that are different from axioms (4)–(11) are the *hypotheses* of C.

A calculation is *bidirectional* if the only inference rules used in it are the replacement rules (1).

For instance, (13) is a bidirectional calculation with the source $F \vee \bot \equiv F$, destination $F \equiv F$, and the empty set of hypotheses.

We will represent calculations as follows: in applications of two-premise rules (1), the horizontal bar will be replaced by $\Leftrightarrow$, and the hint $G \equiv H$ will be written to the right of the arrow in braces $\{\ \}$; in applications of one-premise rules (2) and (3), the horizontal bar will be replaced by $\Rightarrow$. In this notation, bidirectional calculations are distinguished by the absence of the sign $\Rightarrow$ in them. For instance, (13) will be written as

$$\begin{array}{ll} & F \vee \bot \equiv F \\ \Leftrightarrow & \quad \{\ F \vee \bot \equiv F\ \} \\ & F \equiv F\ . \end{array}$$

As another example, here is a calculation with the source $F \vee \forall x G(x)$, destination $F \vee G(t)$, and the empty set of hypotheses; we assume that x is not free in F and that t is free for x in $G(x)$:

$$\begin{array}{ll} & F \vee \forall x G(x) \\ \Leftrightarrow & \quad \{\ F \vee \forall x G(x) \equiv \forall x (F \vee G(x))\ \} \\ & \forall x (F \vee G(x)) \\ \Rightarrow & \\ & F \vee G(t)\ . \end{array}$$

It is clear that if there exists a calculation with source F, destination G, and set of hypotheses Γ, then

$$G \text{ is derivable from } \Gamma \cup \{F\}.$$

If the calculation is bidirectional then we can assert also that

$$F \text{ is derivable from } \Gamma \cup \{G\}$$

and that

$$F \equiv G,\ G \equiv F \text{ are derivable from } \Gamma.$$

To prove the first claim, reverse the order of members in the given calculation; the result will be a calculation again, because every application of any of the two replacement rules will turn into an application of the other replacement rule. To prove the second claim, replace every member H of the calculation with $F \equiv H$; the result will be a calculation whose source

(12) is provable in **DS** and whose destination is $F \equiv G$. The argument for $G \equiv F$ is similar.

Our next goal is to establish the provability of some (classes of) formulas in **DS** by exhibiting in every case an appropriate calculation whose hypotheses are already known to be provable. This will illustrate the usefulness of calculations (and this is also needed for our proof of the completeness of **DS**).

Abbreviations: $\neg F$ stands for $F \equiv \bot$, and $\top$ stands for $\neg\bot$ (that is, $\bot \equiv \bot$). Note that

$$\top \tag{14}$$

is provable in **DS**, because it is an instance of (12).

The theorem schemas that we want to prove are

$$(F \equiv \top) \equiv F, \tag{15}$$

$$\neg\neg F \equiv F, \tag{16}$$

$$(F \equiv F) \equiv \top, \tag{17}$$

$$(F \equiv \neg F) \equiv \bot, \tag{18}$$

$$F \vee \top \equiv \top, \tag{19}$$

$$F \vee \top, \tag{20}$$

$$F \vee \neg F \equiv \top, \tag{21}$$

$$F \vee \neg F, \tag{22}$$

$$(\neg F \equiv G) \equiv \neg(F \equiv G), \tag{23}$$

$$\neg F \vee G \equiv (F \vee G \equiv G), \tag{24}$$

$$\neg(F \vee G) \vee G \equiv \neg F \vee G, \tag{25}$$

$$\neg(F \vee G) \vee \neg G \equiv \neg G, \tag{26}$$

$$(F \equiv G) \equiv (\neg F \vee \neg G \equiv \neg(F \vee G)), \tag{27}$$

$$(F \equiv G) \equiv \neg(F \vee G) \vee \neg(\neg F \vee \neg G). \tag{28}$$

Calculations establishing the provability of four of these schemas are shown below; the other 10 are given in Appendix.

To improve readability, we will replace hint formulas by references explaining why they are provable in **DS**, and we will sometimes combine several derivation steps into one.

Proof of (17):

$$
\begin{array}{ll}
& (F \equiv F) \equiv (\bot \equiv \bot) \\
\Leftrightarrow & \{ \text{ commutativity and associativity of } \equiv \} \\
& (F \equiv \bot) \equiv (F \equiv \bot) \\
\{ \text{ (12) } \} \ .
\end{array}
$$

(The last line is a comment explaining why the destination formula is provable in **DS**. It allows us to conclude that the source formula is provable in **DS** also.)

Proof of (21):

$$
\begin{array}{ll}
& F \vee (F \equiv \bot) \equiv \top \\
\Leftrightarrow & \{ \text{ distributivity of } \vee \text{ over } \equiv \} \\
& (F \vee F \equiv F \vee \bot) \equiv \top \\
\Leftrightarrow & \{ \text{ idempotence of } \vee \text{ and neutrality of } \bot \} \\
& (F \equiv F) \equiv \top \\
\{ \text{ 17) } \} \ .
\end{array}
$$

Proof of (24):

$$
\begin{array}{ll}
& (F \equiv \bot) \vee G \\
\Leftrightarrow & \{ \text{ commutativity of } \vee \text{ and distributivity of } \vee \text{ over } \equiv \} \\
& F \vee G \equiv G \vee \bot \\
\Leftrightarrow & \{ \text{ neutrality of } \bot \} \\
& F \vee G \equiv G \ .
\end{array}
$$

Proof of (25):

$$
\begin{array}{ll}
& \neg(F \vee G) \vee G \\
\Leftrightarrow & \{ \text{ (24) } \} \\
& F \vee G \vee G \equiv G \\
\Leftrightarrow & \{ \text{ associativity and idempotence of } \vee \} \\
& F \vee G \equiv G \\
\Leftrightarrow & \{ \text{ (24) } \} \\
& \neg F \vee G \ .
\end{array}
$$

The last calculation shows how to derive (25) from two instances of "lemma" (24). The calculation before the last tells us how to prove the lemma. In this sense, the last two calculations "encode" a proof of (25) in **DS**. Generally, any proof tree T in **DS** can be represented as a sequence $C_1, \ldots, C_n$ of calculations so that

- the destination of C_n is the end formula of T, and
- for every i $(1 \leq i \leq n)$, the source of C_i is an axiom, and the hypotheses of C_i are among the destinations of $C_1, \ldots, C_{i-1}$.

This is easy to check by induction on the size of T.

In this sense, all proofs in the formalization of predicate logic introduced above can be reduced to calculations.

4 Implication

We will use $F \supset G$ as shorthand for $\neg F \vee G$. For instance,

$$F \supset (F \vee G) \tag{29}$$

stands for $\neg F \vee (F \vee G)$; it is easy to see that this formula is provable in **DS**.

Three topics related to implication will be briefly discussed here: modus ponens, the deduction theorem, and "ping-pong arguments."

The modus ponens rule

$$\frac{F \quad F \supset G}{G} \tag{30}$$

is derivable[3] in **DS** (see Appendix). Consequently, the deductive system **DS*** obtained from **DS** by adding (30) to the set of inference rules has the same theorems as **DS**.

Some examples of calculational proofs from [Dijkstra and Scholten, 1990] can be conveniently explained in terms of the extended system **DS***. The definition of a calculation and the other definitions introduced in the first paragraph of Section 3 apply to the extended system without any changes; the only difference is that now hints are not necessarily equivalences—they may also have the form $F \supset G$. An application of rule (30) in a calculation will be represented by the symbol $\Rightarrow$ with the hint written to the right of the arrow, for instance:

$$\begin{array}{ll} & F \vee \bot \\ \Leftrightarrow & \{ \text{ neutrality of } \bot \ \} \\ & F \\ \Rightarrow & \{ \ (29) \ \} \\ & F \vee G \ . \end{array} \tag{31}$$

[3]An inference rule R is *derivable* in a deductive system Σ if the conclusion (of any instance) of R is derivable from its premises in Σ.

Just as in the less general setting of Section 3, if there exists a calculation with the source F, the destination G, and the set of hypotheses Γ, then G is derivable from $\Gamma \cup \{F\}$ (in $\mathbf{DS}^*$, and consequently in $\mathbf{DS}$). The deduction theorem stated below shows that, under an additional assumption related to the use of the generalization rule, we can claim also that $F \supset G$ is derivable from Γ.

For any calculation C, the variables x from the conclusions $\forall x F$ of all applications of rule (2) in C are said to be *varied* in C. According to the deduction theorem, *if there exists a calculation C with the source F, the destination G, and the set of hypotheses Γ such that the variables varied in C are not free in F, then $F \supset G$ is derivable from Γ.* The proof is given in Appendix.

For instance, according to the deduction theorem, calculation (31) establishes the provability of $(F \vee \bot) \supset (F \vee G)$.

"Ping-pong arguments" correspond to the uses of the rule

$$\frac{F \supset G \quad G \supset F}{F \equiv G}\,, \tag{32}$$

which is derivable in $\mathbf{DS}$ also (see Appendix). The methodological advice given in [Dijkstra and Scholten, 1990] is "to avoid the ping-pong argument if you can, but to learn to recognize the situations in which it is appropriate" (p. 24). The derivability of (32) shows that, in principle, such arguments can be always eliminated.

5 Comparison with Related Work

The discussion of predicate logic in [Dijkstra and Scholten, 1990] aims at developing a methodology that can be used by the working mathematician (or computer scientist) for discovering and presenting proofs. Our goal in this note is to provide a formal analysis of several aspects of this methodology. Many useful proposals found in the book are not reflected in our discussion of calculations.

We can mention two notational conventions as examples. One is to drop parentheses in multiple equivalences. For instance, Dijkstra and Scholten would write (27) as

$$F \equiv G \equiv \neg F \vee \neg G \equiv \neg(F \vee G)$$

and then reference this expression in a hint no matter whether what is actually needed is (25), or

$$((F \equiv G) \equiv (\neg F \vee \neg G)) \equiv \neg(F \vee G),$$

or

$$(F \equiv (G \equiv (\neg F \vee \neg G))) \equiv \neg(F \vee G),$$

etc. (All these formulas follow from each other by the associativity of equivalence.) The other convention allows us—in the terminology of this note—to reverse the order of members in the derivations that are not bidirectional, so that the source appears at the end and the destination at the beginning, provided that each $\Rightarrow$ is simultaneously replaced with $\Leftarrow$. For instance, (31) can be written also as

$$\begin{array}{ll} & F \vee G \\ \Leftarrow & \quad \{ \text{ (29) } \} \\ & F \\ \Leftrightarrow & \quad \{ \text{ neutrality of } \perp \} \\ & F \vee \perp . \end{array}$$

Gries and Schneider [1995] define a deductive system that can be used to formalize calculational proofs in propositional logic. Instead of our replacement rules (1), their system **E** includes the rules

$$\frac{G \equiv H}{F(G) \equiv F(H)}\,, \qquad \frac{F \equiv G \quad G \equiv H}{F \equiv H}\,, \qquad \frac{F \quad F \equiv G}{G}\,.$$

(It includes also a rule for substituting formulas for propositional variables and uses individual axioms instead of axiom schemas.) The authors prove the completeness of **E** and show, by means of examples, how "calculational proofs" can be formalized in **E**.[4]

The analysis of the calculational style proposed in this note extends the contribution of [Gries and Schneider, 1995] in three ways. First, we gave a formal definition of a calculation. Second, we showed how to represent any proof in a certain sound and complete deductive system as a sequence of calculations. Third, our theory is not restricted to the propositional case.

[4] A detailed discussion of this system can be found in [Bijlsma and Nederpelt, 1998]. Its earlier versions are described in [Gries, 1981] and [Gries and Schneider, 1993]. The first of the three inference rules above is mentioned in [Baaz and Zach, 1994].

Appendix: Proofs

Proofs of Formulas from Section 3

Proof of (15):

$$
\begin{array}{ll}
 & (F \equiv (\perp \equiv \perp)) \equiv F \\
\Leftrightarrow & \{ \text{ commutativity and associativity of } \equiv \} \\
 & (F \equiv \perp) \equiv (F \equiv \perp) \\
\{ (12) \} \ . &
\end{array}
$$

Proof of (16):

$$
\begin{array}{ll}
 & ((F \equiv \perp) \equiv \perp) \equiv F \\
\Leftrightarrow & \{ \text{ commutativity and associativity of } \equiv \} \\
 & (F \equiv \perp) \equiv (F \equiv \perp) \\
\{ (12) \} \ . &
\end{array}
$$

Proof of (18):

$$
\begin{array}{ll}
 & (F \equiv (F \equiv \perp)) \equiv \perp \\
\Leftrightarrow & \{ \text{ commutativity and associativity of } \equiv \} \\
 & (F \equiv \perp) \equiv (F \equiv \perp) \\
\{ (12) \} \ . &
\end{array}
$$

Proof of (19):

$$
\begin{array}{ll}
 & F \vee (\perp \equiv \perp) \equiv (\perp \equiv \perp) \\
\Leftrightarrow & \{ \text{ distributivity of } \vee \text{ over } \equiv \} \\
 & (F \vee \perp \equiv F \vee \perp) \equiv (\perp \equiv \perp) \\
\Leftrightarrow & \{ \text{ commutativity and associativity of } \equiv \} \\
 & (F \vee \perp \equiv \perp) \equiv (F \vee \perp \equiv \perp) \\
\{ (12) \} \ . &
\end{array}
$$

Proof of (20):

$$
\begin{array}{ll}
 & F \vee \top \\
\Leftrightarrow & \{ (15) \} \\
 & F \vee \top \equiv \top \\
\{ (19) \} \ . &
\end{array}
$$

Proof of (22):

$$
\begin{array}{ll}
 & F \vee \neg F \\
\Leftrightarrow & \{ (15) \} \\
 & F \vee \neg F \equiv \top \\
\{ (21) \} \ . &
\end{array}
$$

Proof of (23):

$$
\begin{array}{ll}
 & (F \equiv \bot) \equiv G \\
\Leftrightarrow & \{ \text{ commutativity and associativity of } \equiv \} \\
 & (F \equiv G) \equiv \bot \ .
\end{array}
$$

Proof of (26):

$$
\begin{array}{ll}
 & \neg(F \vee G) \vee \neg G \\
\Leftrightarrow & \{ \ (24) \ \} \\
 & F \vee G \vee \neg G \equiv \neg G \\
\Leftrightarrow & \{ \text{ associativity of } \vee \text{ and } (21) \ \} \\
 & F \vee \top \equiv \neg G \\
\Leftrightarrow & \{ \ (19) \ \} \\
 & \top \equiv \neg G \\
\Leftrightarrow & \{ \text{ commutativity of } \equiv \text{ and } (15) \ \} \\
 & \neg G \ .
\end{array}
$$

Proof of (27):

$$
\begin{array}{ll}
 & \neg F \vee \neg G \equiv \neg(F \vee G) \\
\Leftrightarrow & \{ \text{ commutativity of equivalence and } (23) \ \} \\
 & \neg(F \vee G \equiv \neg F \vee \neg G) \\
\Leftrightarrow & \{ \ (24) \ \} \\
 & \neg(F \vee G \equiv (F \vee \neg G \equiv \neg G)) \\
\Leftrightarrow & \{ \text{ commutativity and associativity of equivalence, } (23) \text{ and } (16) \ \} \\
 & (F \vee G \equiv F \vee \neg G) \equiv G \\
\Leftrightarrow & \{ \text{ distributivity of } \vee \text{ over } \equiv \} \\
 & F \vee (G \equiv \neg G) \equiv G \\
\Leftrightarrow & \{ \ (18) \text{ and neutrality of } \bot \ \} \\
 & F \equiv G \ .
\end{array}
$$

Proof of (28):

$$
\begin{array}{ll}
 & \neg(F \vee G) \vee \neg(\neg F \vee \neg G) \\
\Leftrightarrow & \{ \text{ commutativity of } \vee \text{ and } (24) \ \} \\
 & \neg F \vee \neg G \vee \neg(F \vee G) \equiv \neg(F \vee G) \\
\Leftrightarrow & \{ \text{ associativity of } \vee \text{ and } (26) \ \} \\
 & \neg F \vee \neg G \equiv \neg(F \vee G) \\
\Leftrightarrow & \{ \ (27) \ \} \\
 & F \equiv G \ .
\end{array}
$$

Derivable Inference Rules

The derivability of (30) is demonstrated by the calculation

$$
\begin{array}{ll}
 & \neg F \vee G \\
\Leftrightarrow & \quad \{ (24) \} \\
 & F \vee G \equiv G \\
\Leftrightarrow & \quad \{ F \equiv \top \} \\
 & \top \vee G \equiv G \\
\Leftrightarrow & \quad \{ \text{commutativity of } \equiv \text{ and } \vee \text{ and } (19) \} \\
 & G \equiv \top \\
\Leftrightarrow & \quad \{ (15) \} \\
 & G
\end{array}
$$

in view of the fact that the hint $F \equiv \top$ can be derived from F using (15).

The derivability of (32) is demonstrated by the calculation

$$
\begin{array}{ll}
 & \neg F \vee G \\
\Leftrightarrow & \quad \{ (24) \text{ and commutativity of disjunction} \} \\
 & G \vee F \equiv G \\
\Leftrightarrow & \quad \{ G \vee F \equiv F \} \\
 & F \equiv G
\end{array}
$$

in view of the fact that the hint $G \vee F \equiv F$ can be derived from $\neg G \vee F$ using (24).

The proof of the deduction theorem below uses the derivability of the rules

$$\frac{F \supset G \quad G \supset H}{F \supset H}\ , \tag{33}$$

$$\frac{F \supset G(x)}{F \supset \forall x G(x)} \quad \text{where } x \text{ is not free in } F \tag{34}$$

and

$$\frac{F(x) \supset \forall x G(x)}{F(x) \supset G(t)} \quad \text{where } t \text{ is free for } x \text{ in } G(x). \tag{35}$$

In view of (24), the derivability of (33) will be proved if we establish the derivability of the rule

$$\frac{F \vee G \equiv G \quad G \vee H \equiv H}{F \vee H \equiv H}\ .$$

This is demonstrated by the calculation

$$\begin{array}{ll} & F \vee H \\ \Leftrightarrow & \{ G \vee H \equiv H \} \\ & F \vee (G \vee H) \\ \Leftrightarrow & \{ \text{associativity of } \vee \text{ and } F \vee G \equiv G \} \\ & G \vee H \\ \Leftrightarrow & \{ G \vee H \equiv H \} \\ & H \ . \end{array}$$

The derivability of (34) is demonstrated by the calculation

$$\begin{array}{ll} & \neg F \vee G(x) \\ \Rightarrow & \\ & \forall x(\neg F \vee G(x)) \\ \Leftrightarrow & \{ \vee \text{ commutes with } \forall \} \\ & \neg F \vee \forall x G(x) \ . \end{array}$$

The derivability of (35) is demonstrated by the calculation

$$\begin{array}{ll} & \neg F(x) \vee \forall x G(x) \\ \Rightarrow & \\ & \forall x(\neg F(x) \vee \forall x G(x)) \\ \Rightarrow & \\ & \neg F(y) \vee \forall x G(x) \\ \Leftrightarrow & \{ \vee \text{ commutes with } \forall \} \\ & \forall x(\neg F(y) \vee G(x)) \\ \Rightarrow & \\ & \neg F(y) \vee G(t) \\ \Rightarrow & \\ & \forall y(\neg F(y) \vee G(t)) \\ \Rightarrow & \\ & \neg F(x) \vee G(t) \end{array}$$

where y is a variable that is not free in $F(x)$, $G(x)$ and t.

The proof of the completeness of **DS** below uses the derivability of the rule

$$\frac{F \equiv G}{\forall x F \equiv \forall x G} \ . \qquad (36)$$

To prove that this rule is derivable, note first that the formula

$$\forall x F \supset \forall x G$$

can be derived from the premise $F \equiv G$ of (36) as follows:

$$\begin{array}{l}
\{\ (22)\ \} \\
\quad \forall xF \vee \neg\forall xF \\
\Leftrightarrow \quad \{\ \vee \text{ is commutative and commutes with } \forall\ \} \\
\quad \forall x(\neg\forall xF \vee F) \\
\Rightarrow \\
\quad \neg\forall xF \vee F \\
\Leftrightarrow \quad \{\ F \equiv G\ \} \\
\quad \neg\forall xF \vee G \\
\Rightarrow \\
\quad \forall x(\neg\forall xF \vee G) \\
\Leftrightarrow \quad \{\ \vee \text{ commutes with } \forall\ \} \\
\quad \neg\forall xF \vee \forall xG\ .
\end{array}$$

The formula

$$\forall xG \supset \forall xF$$

can be derived from the premise (36) in a similar way. Now the derivability of the conclusion of (36) from its premise follows using the derivability of (32).

Proof of the Deduction Theorem

Proposition. *If there exists a calculation C in $\mathbf{DS}^*$ with the source F, the destination G, and the set of hypotheses Γ such that the variables varied in C are not free in F, then $F \supset G$ is derivable from Γ.*

Replace every member H of C with $F \supset H$. The source of C turns into $F \supset F$, which equals $\neg F \vee F$ and follows from (22) by the commutativity of disjunction. Applications of the replacement rules (1) turn into applications of the same rules. Applications of (2) turn into applications of (34). Applications of (3) turn into applications of (35). The destination turns into $F \supset G$.

Proof of the Completeness Theorem

Proposition. *Every logically valid formula is provable in* **DS**.

We will prove this fact by relating **DS** to another formalization of predicate logic, which is defined and proved to be complete, for instance, in Chapter 2 of [Mendelson, 1987]. The formulas of that other system **K** are built from atomic formulas using negation, implication and the universal quantifier. In the description of **K** below we represent negation and implication by the characters $\sim$ and $\to$, to distinguish it from the symbols $\neg$ and $\supset$ introduced in Sections 3 and 4 as abbreviations.

The axiom schemas of **K** are

$$F \to (G \to F), \tag{37}$$

$$(F \to (G \to H)) \to ((F \to G) \to (F \to H)), \tag{38}$$

$$({\sim}F \to {\sim}G) \to (({\sim}F \to G) \to F), \tag{39}$$

$$\forall x F(x) \to F(t), \text{ where } t \text{ is free for } x \text{ in } F(x), \tag{40}$$

$$\forall x(F \to G(x)) \to (F \to \forall x G(x)), \text{ where } x \text{ is not free in } F. \tag{41}$$

The inference rules of **K** are

$$\frac{F \quad F \to G}{G} \tag{42}$$

and the generalization rule (2)—with F understood to stand for a formula of the language of **K**.

To prove the completeness of **DS**, we will define a translation α from the language of **DS** into the language of **K** and a translation β from the language of **K** into the language of **DS**, and will prove the following claims about them:

Claim 1. If F is a logically valid formula in the language of **DS** then αF is logically valid also.

Claim 2. If F is provable in **K** then βF is provable in **DS**.

Claim 3. For any formula F in the language of **DS**, $\beta\alpha F \equiv F$ is provable in **DS**.

Since **K** is complete, the completeness of **DS** follows from Claims 1–3.

The translation α is defined as follows:

1. $\alpha F = F$ if F is atomic,
2. $\alpha\bot = {\sim}(P \to P)$ for some propositional symbol P,
3. $\alpha(F \equiv G) = (\alpha F \to {\sim}\alpha G) \to {\sim}({\sim}\alpha F \to \alpha G)$,
4. $\alpha(F \vee G) = {\sim}\alpha F \to \alpha G$,
5. $\alpha(\forall x F) = \forall x \alpha F$.

Here is the definition of β:

1. $\beta F = F$ if F is atomic,
2. $\beta(\sim F) = \neg\beta F$,
3. $\beta(F \to G) = \neg\beta F \vee \beta G$,
4. $\beta(\forall x F) = \forall x \beta F$.

Claim 1 follows from the fact that αF is semantically equivalent to F, which is easy to check by structural induction.

To verify Claim 2, we will first prove in **DS** the counterparts of axioms (37)–(41), that is, the formulas

$$F \supset (G \supset F), \tag{43}$$

$$(F \supset (G \supset H)) \supset ((F \supset G) \supset (F \supset H)), \tag{44}$$

$$(\neg F \supset \neg G) \supset ((\neg F \supset G) \supset F), \tag{45}$$

$$\forall x F(x) \supset F(t), \text{ where } t \text{ is free for } x \text{ in } F(x), \tag{46}$$

$$\forall x(F \supset G(x)) \supset (F \supset \forall x G(x)), \text{ where } x \text{ is not free in } F. \tag{47}$$

Proof of (43):

$$\begin{array}{ll} & \neg F \vee (\neg G \vee F) \\ \Leftrightarrow & \{ \text{ commutativity and associativity of } \vee \} \\ & \neg G \vee (F \vee \neg F) \\ \Leftrightarrow & \{ \text{ (21) } \} \\ & \neg G \vee \top \\ \{ \text{ (20) } \} \ . & \end{array}$$

Proof of (44):

$$\begin{array}{ll} & \neg(\neg F \vee (\neg G \vee H)) \vee (\neg(\neg F \vee G) \vee (\neg F \vee H)) \\ \Leftrightarrow & \{ \text{ commutativity and associativity of } \vee \text{ and (25) } \} \\ & \neg(\neg F \vee (\neg G \vee H)) \vee \neg G \vee \neg F \vee H \\ \Leftrightarrow & \{ \text{ commutativity and associativity of } \vee \text{ and (25) } \} \\ & \neg(\neg G \vee H) \vee \neg G \vee \neg F \vee H \\ \Leftrightarrow & \{ \text{ commutativity of } \vee \text{ and (25) } \} \\ & \neg H \vee \neg G \vee \neg F \vee H \\ \Leftrightarrow & \{ \text{ commutativity and associativity of } \vee \text{ and (21) } \} \\ & \neg G \vee \neg F \vee \top \\ \{ \text{ (20) } \} \ . & \end{array}$$

Proof of (45):

$$
\begin{array}{ll}
& \neg(\neg\neg F \vee \neg G) \vee (\neg(\neg\neg F \vee G) \vee F) \\
\Leftrightarrow & \{\ (16)\ \} \\
& \neg(F \vee \neg G) \vee (\neg(F \vee G) \vee F) \\
\Leftrightarrow & \{\ \text{commutativity and associativity of } \vee \text{ and (25)}\ \} \\
& (F \vee \neg G) \vee \neg(F \vee \neg G) \\
& \{\ (22)\ \}\ .
\end{array}
$$

Proof of (46):

$$
\begin{array}{ll}
& \{\ (22)\ \} \\
& \forall x F(x) \vee \neg\forall x F(x) \\
\Leftrightarrow & \{\ \vee \text{ is commutative and commutes with } \forall\ \} \\
& \forall x(\neg\forall x F(x) \vee F(x)) \\
\Rightarrow & \\
& \neg\forall x F(x) \vee F(t)\ .
\end{array}
$$

Proof of (47):

$$
\begin{array}{ll}
& \neg\forall x(\neg F \vee G(x)) \vee (\neg F \vee \forall x G(x)) \\
\Leftrightarrow & \{\ \vee \text{ commutes with } \forall \text{ and is commutative}\ \} \\
& \forall x(\neg F \vee G(x)) \vee \neg\forall x(\neg F \vee G(x)) \\
& \{\ (22)\ \}\ .
\end{array}
$$

Now we can prove Claim 2 by induction on the proof of F in **K**. The translation β maps the axioms (37)–(41) of **K** to instances of the corresponding theorem schemas (43)–(47). Furthermore, it maps any instance of rule (42) to an instance of the derivable rule (30), and it maps any instance of rule (2) to an instance of the same rule in the language of **DS**.

Claim 3 is proved by structural induction. If F is an atomic formula then $\beta\alpha F = F$, so that $\beta\alpha F \equiv F$ is an instance of (12). If F is $\perp$ then $\beta\alpha F$ is $\neg(\neg P \vee P)$, so that the equivalence to be proved equals $\neg\neg(\neg P \vee P)$. This formula follows from (22) using (16) and the commutativity of disjunction. Assume now that $\beta\alpha F \equiv F$ and $\beta\alpha G \equiv G$ are provable in **DS**. We need to verify that

$$\beta\alpha(F \vee G) \equiv F \vee G \tag{48}$$

and

$$\beta\alpha(F \equiv G) \equiv (F \equiv G) \tag{49}$$

are provable in **DS** also. Proof of (48):

$$
\begin{array}{ll}
& \beta\alpha(F \vee G) \\
= & \quad \{ \text{ definition of } \alpha \ \} \\
& \beta({\sim}\,\alpha F \to \alpha G) \\
= & \quad \{ \text{ definition of } \beta \ \} \\
& \neg\neg\beta\alpha F \vee \beta\alpha G \\
\Leftrightarrow & \quad \{ \text{ induction hypothesis } \} \\
& \neg\neg F \vee G \\
\Leftrightarrow & \quad \{ \ (16) \ \} \\
& F \vee G \ .
\end{array}
$$

(The expressions connected with the equal sign represent the same formula; these transitions do not correspond to derivation steps.) Proof of (49):

$$
\begin{array}{ll}
& \beta\alpha(F \equiv G) \\
= & \quad \{ \text{ definition of } \alpha \ \} \\
& \beta((\alpha F \to {\sim}\,\alpha G) \to {\sim}({\sim}\,\alpha F \to \alpha G)) \\
= & \quad \{ \text{ definition of } \beta \ \} \\
& \neg(\neg\beta\alpha F \vee \neg\beta\alpha G) \vee \neg(\neg\neg\beta\alpha F \vee \beta\alpha G) \\
\Leftrightarrow & \quad \{ \text{ induction hypothesis } \} \\
& \neg(\neg F \vee \neg G) \vee \neg(\neg\neg F \vee G) \\
\Leftrightarrow & \quad \{ \ (16) \text{ and commutativity of } \vee \ \} \\
& \neg(F \vee G) \vee \neg(\neg F \vee \neg G) \\
\Leftrightarrow & \quad \{ \ (28) \ \} \\
& F \equiv G \ .
\end{array}
$$

Finally, assuming that $F \equiv \beta\alpha F$ is provable in **DS**, we need to check that $\forall x F \equiv \beta\alpha\forall x F$ is provable in **DS** also. The last formula can be written as $\forall x F \equiv \forall x \beta\alpha F$; consequently, it follows by one application of (36).

Acknowledgements

I would like to thank Robert Boyer, Edsger W. Dijkstra, W. H. J. Feijen, Michael Gelfond, Lew Gordeew, David Gries, Norman McCain, Grigori Mints, Jay Misra, Rob Nederpelt, Vladimir Orevkov and Armando Tacchella for comments related to the subject of this note. This work was partially supported by National Science Foundation under grant IIS-9732744.

References

[Baaz and Zach, 1994] Matthias Baaz and Richard Zach. Short proofs of tautologies using the schema of equivalence. In *Computer Science Logic: Seventh Workshop, CSL'93 (Lecture Notes in Computer Science 832)*, pages 33–35. Springer-Verlag, 1994.

[Bijlsma and Nederpelt, 1998] Lex Bijlsma and Rob Nederpelt. Dijkstra–scholten predicate calculus: concepts and misconceptions. *Acta Informatica*, 35:1007–1036, 1998.

[Dijkstra and Scholten, 1990] Edsger Dijkstra and Carel Scholten. *Predicate Calculus and Program Semantics.* Springer-Verlag, 1990.

[Gries and Schneider, 1993] David Gries and Fred Schneider. *A Logical Approach to Discrete Math.* Springer-Verlag, 1993.

[Gries and Schneider, 1995] David Gries and Fred Schneider. Equational propositional logic. *Information Processing Letters*, 53:145–152, 1995.

[Gries, 1981] David Gries. *The Science of Programming.* Springer-Verlag, 1981.

[Mendelson, 1987] Elliott Mendelson. *Introduction to Mathematical Logic.* Wadsworth & Brooks, 1987. Third edition.