International Journal of Applied Research 2018; 4(1): 471-476

ISSN Print: 2394-7500
ISSN Online: 2394-5869
Impact Factor: 5.2
IJAR 2018; 4(1): 471-476
www.allresearchjournal.com
Received: 26-11-2017
Accepted: 28-12-2017

Dr. Daya Shankar Pratap
Research Scholar, Department of Mathematics, JP University, Chapra, Bihar, India

Corresponding Author:
Dr. Daya Shankar Pratap
Research Scholar, Department of Mathematics, JP University, Chapra, Bihar, India

3 Analysis of Boolean algebra to unified algebra

Dr. Daya Shankar Pratap

Abstract
This paper is about the symbols and notations of boolean algebra, and about the way the subject is explained. It is about education, and about putting boolean algebra into general use and practice. To make the scope clear, by "boolean algebra" I mean the algebra whose expressions are of type boolean. I mean to include the expressions of propositional calculus and predicate calculus. I shall say "boolean algebra" or "boolean calculus" interchangeably, and call the expressions of this algebra "boolean expressions". Analogously, I say "number algebra" or "number calculus" interchangeably, and call the expressions of that algebra "number expressions".

Keywords: Boolean algebra and unified algebra

Introduction
Boolean algebra is the basic algebra for much of computer science. Other applications include digital circuit design, law, reasoning about any subject, and any kind of specifications, as well as providing a foundation for all of mathematics. Boolean algebra is inherently simpler than number algebra. There are only two boolean values and a few boolean operators, and they can be explained by a small table. There are infinitely many number values and number operators, and even the simplest, counting, is inductively defined. So why is number algebra taught in primary school, and boolean algebra in university? Why isn't boolean algebra better known, better accepted, and better used?
One reason may be that, although boolean algebra is just as useful as number algebra, it isn't as necessary. Informal methods of reckoning quantity became intolerable several thousand years ago, but we still get along with informal methods of specification, design, and reasoning. Another reason may be just an accident of educational history, and still another may be our continuing mistreatment of boolean algebra.

Boolean calculation
Given an expression, it is often useful to find an equivalent but simpler expression. For example, in number algebra

$x\times(z+l) - y\times(z-1) - z\times(x-y)$ distribute
$= (x\times z + x\times 1) - (y\times z - y\times l) - (z\times x - z\times y)$ unity and double negation
$= x\times z + x - y\times z + y - z\times x + z\times y$ symmetry and associativity
$= x + y + (x\times z - x\times z) + (y\times z - y\times z)$ zero and identity
$= x+y$

We might sometimes want to find an equivalent expression that isn't simpler; to remove the directionality I'll say "calculation" rather than "simplification". We can use operators other than = down the left side of the calculation; we can even use a mixture of operators, as long as there is transitivity. For example, the calculation (for real x)

$x\times(x + 2)$ distribute
$= x^2 + 2\times x$ add and subtract 1
$= x^2+2\times x+ 1- 1$ factor
$= (x + 1)^2 - 1$ a square is nonnegative
≥ -1

tells us

$$x\times(x + 2)\geq -1$$

Boolean calculation is similar. For example,

$(a \Rightarrow b) \vee (b \Rightarrow a)$ replace implications
$= \neg a \vee b \vee \neg b \neg a$ $\vee$ is symmetric
$= a \vee \neg a \vee b \vee \neg b$ excluded middle, twice
$=$ true $\vee$ true V is idempotent
$=$ true

And so $(a \Rightarrow b) \vee (b \Rightarrow a)$ has been simplified to true, which is to say it has been proven. Here is another example.

$\backslash n \cdot n + n^2 = n^3$ instance
$\Leftarrow \quad 0 + 0^2 = 0^3$ arithmetic
$=$ true

And so $(\backslash n \cdot n + n^2 = n^2) \Leftarrow$ true, and so $\backslash n \cdot n + n^2 = n^3$ is proven.

Solving simultaneous equations can also be done as a boolean calculation. For example, $x+x \times y+y=5 \wedge x-x \times y+y=1$ subtract and add $2 \times x \times y$ in first equation

$= \quad x-x \times y+y+2 \times x \times y=5 \wedge x-x \times y+y=1$
use second equation to simplify first
$= \quad 1+2 \times x \times y=5 \wedge x-x \times y+y=1$
$= \quad 2 \times x \times y=4 \wedge x-x \times y+y=1$
$= \quad x \times y=2 \wedge x-x \times y+y=1$
use first equation to simplify second
$= \quad x \times y=2 \wedge x-2+y=l$
$= \quad x \times y=2 \wedge x+y=3$
$= \quad x=l \wedge y=2 \vee x=2 \wedge y=l$
$\Leftarrow \quad x=l \wedge y=2$

These examples show that simplifying, proving, and solving are all the same: they are all just calculation.

Traditional notations

Arithmetic notations are reasonably standard throughout the world. The expression 738 + 45 = 783 is recognized and understood by schoolchildren almost everywhere. But there are no standard boolean notations.

Even the two boolean constants have no standard symbols. Symbols in use include

true	t	tt	T	1	0	1 = 1
false	f	ff	F	0	1	1 = 2

Quite often the boolean constants are written as 1 and 0, with + for disjunction, juxtaposition for conjunction, and perhaps - for negation. With this notation, here are some laws.

$x(y+z) = xy + xz$
$x + yz = (x+y)(x+z)$
$x+-x=l$
$x(-x) = 0$

The first law above coincides with number algebra, but the next three clash with number algebra. The near-universal reaction of algebraists to notational criticisms is: it doesn't matter which symbols are used; just introduce them, and get on with it.

The most common notations for the two boolean constants found in programming languages and in programming textbooks seem to be true and false. I have two objections to these symbols. The first is that they are English-based and clumsy. Number algebra could never have advanced to its present state if we had to write out words for numbers. seven three eight + four five = seven eight three is just too clumsy, and so is

true $\wedge$ false $\vee$ true $\equiv$ true

Clumsiness may seem minor, but it can be the difference between success and failure in a calculus.

For symbols that are independent of the application, I propose the lattice symbols a and b, pronounced "top" and "bottom". Since Boolean algebra is the mother of all lattices, I think it is appropriate, not a misuse of those symbols. They can equally well be used for true and false statements, for high and low voltages (power and ground), for satisfactory and unsatisfactory tables, for innocent and guilty behavior, or any other opposites.

For disjunction, the symbol V is fairly standard, coming from the Latin word "vel" for "or". For conjunction, the symbol is less standard, the two most common choices being & and A. We are even less settled on a symbol for implication. Symbols in use include

$\rightarrow \Rightarrow \therefore \supset$

The usual explanation says it means "if then", followed by a discussion about the meaning of "if then". Apparently, people find it difficult to understand an implication whose antecedent is false ; for example, "If my mother had been a man, I'd be the king of France."

From Booleans to numbers

Some boolean expressions are laws: they have value a no matter what values are assigned to the variables. Some boolean expressions are unsatisfiable: they have value b no matter what values are assigned to the variables. The remaining boolean expressions are in between, and "solving" means finding an assignment of values for the variables for which the boolean expression has value a. (Solving is not just for equations but for any kind of boolean expression.) A lot of mathematics is concerned with solving. And in particular, number algebra has developed by the desire to solve. To caricature the development, we choose an unsatisfiable boolean expression and say, "What a pity that it has no solutions. Let's give it one." This has resulted in an increasing sequence of domains, from naturals to integers to rationals to reals to complex numbers. The boolean expression x+ 1 = 0 is unsatisfiable in the natural numbers, but we give it a solution and thereby invent the integers. Similarly we choose to give solutions to $x \times 2=1$, $x^2=2$, $x^2 = -1$, and thereby progress to larger domains. This progression is both historical and pedagogical. At the same time as we gain solutions, we lose laws, since the laws and unsatisfiable expressions are each other's negations. For example, when we gain a solution to $x^2= 2$, we lose the law $x^2 \neq 2$.

As the domain of an operation or function grows, we do not change its symbol; addition is still denoted + as we go from naturals to complex numbers.

Historically, number algebra did not grow from boolean algebra; but pedagogically it can do so. As already argued, the use of 0 1 + × for b a $\vee$ $\wedge$ doesn't work. To find an association between booleans and numbers that works for unification, we must use a number system extended with an infinite number. Such a system is useful for many purposes; for example, it is used to prove things about the execution time of programs (some execution times are infinite). For a list of axioms of this arithmetic. The association that works is as follows.

boolean			number
top	$\top$	∞	infinity
bottom	$\bot$	$-\infty$	minus infinity
negation	$\neg$	$-$	negation
conjunction	$\wedge$	$\downarrow$	minimum
disjunction	$\vee$	$\uparrow$	maximum
implication	$\Rightarrow$	$\leq$	order
equivalence	$\equiv$	$=$	equality
exclusive or	$\oplus$	$\neq$	inequality

With this association, all number laws employing only these operators correspond to boolean laws. For example,

boolean law	number law
$\top \equiv \neg\bot$	$\infty = - -\infty$
$a \equiv \neg\neg a$	$x = - -x$
$a \vee \top \equiv \top$	$x \uparrow \infty = \infty$
$a \wedge \bot \equiv \bot$	$x \downarrow -\infty = -\infty$
$a \vee \bot \equiv a$	$x \uparrow -\infty = x$
$a \wedge \top \equiv a$	$x \downarrow \infty = x$
$a \Rightarrow \top$	$x \leq \infty$
$\bot \Rightarrow a$	$-\infty \leq x$
$a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$	$x \uparrow (y \downarrow z) = (x \uparrow y) \downarrow (x \uparrow z)$
$a \wedge (b \vee c) \equiv (a \wedge b) \vee (a \wedge c)$	$x \downarrow (y \uparrow z) = (x \downarrow y) \uparrow (x \downarrow z)$
$a \vee b \equiv \neg(\neg a \wedge \neg b)$	$x \uparrow y = -(-x \downarrow -y)$
$a \wedge b \equiv \neg(\neg a \vee \neg b)$	$x \downarrow y = -(-x \uparrow -y)$

There are boolean laws that do not correspond to number laws, just as there are integer laws that are not real laws. That's another way of saying that there are unsatisfiable boolean expressions that correspond to satisfiable number expressions. We will use this for our unified development.

Unified algebra

Here is my proposal for the symbols of a unified algebra.

unified		
top	$\top$	infinity
bottom	$\bot$	minus infinity
negation	$-$	negation
conjunction	$\wedge$	minimum
disjunction	$\vee$	maximum
"nand"	Δ	negation of minimum
"nor"	∇	negation of maximum
implication	$\leq$	order
reverse implication	$\geq$	reverse order
strict implication	$<$	strict order
strict reverse implication	$>$	strict reverse order
equivalence	$=$	equality
exclusive or	$\neq$	inequality

The symbols – ≤ ≥ < > = are world-wide standards, used by school children in all countries, so I dare not suggest any change to them. The symbol ≠ for inequality is the next best known, but I have dared to stand up the slash so that all symmetric operators have symmetric symbols and all asymmetric operators have asymmetric symbols. (Although it was not a consideration, ≠ also looks more like ⊕.) The "nand" symbol is a combination of the "not" and "and" symbols, and similarly for "nor". But I am worried that A and V are poor choices because they point the wrong way to be minimum and maximum; it might be better to use ↓ and ↑ for conjunction and disjunction, and ↓ and ↑ for "nand" and "nor". One suggestion: note that ∨ is wide at the top, and ∧ is narrow at the top. Another suggestion: note that ∨ holds water, and ∧ doesn't.

Duality has been sacrificed to standards; the pair ≤ < are duals, so they ought to be vertical reflections of each other; similarly the pair ≥ >, and also = #; addition and subtraction are self-dual, and happily + and – are vertically symmetric; multiplication is not self-dual, but × is unfortunately vertically symmetric.

Having unified the symbols, I suppose we should also unify the terminology. I vote for the number terminology in the right column, except that I prefer to call a and b "top" and "bottom".

The association between booleans and numbers suggested here allows the greatest number of boolean laws to be generalized to all numbers. For example, if a, b, and c are boolean, it is usual to define if a then b else c as follows:

$$(\text{if a then b else c}) = (a \wedge b) \vee (-a \wedge c)$$

If a remains boolean but b and c are numbers, the if-expression on the left is still sensible (the Algol if), and furthermore it is still equal to the expression on the right. This generalization requires the particular association between booleans and numbers suggested here.

The next examples, written in boolean notations, are the laws

$$(a \wedge b \Rightarrow c) \equiv (a \Rightarrow c) \vee (b \Rightarrow c)$$
$$(a \vee b \Rightarrow c) \equiv (a \Rightarrow c) \vee (b \Rightarrow c)$$

A common error is to use conjunction twice, or disjunction twice. The boolean reading "a and b implies c if and only if a implies c or b implies c" sounds no more reasonable than "a and b implies c if and only if a implies c and b implies c". In unified notation,

$$(a \wedge b \leq c) = (a \leq c) \vee (b \leq c)$$
$$(a \vee b \Rightarrow c) = (a \leq c) \wedge (b \leq c)$$

it is more obvious that the minimum of a and b is less than or equal to c when at least one of a or b is less than or equal to c, and the maximum of a and b is less than or equal to c when both a and b are less than or equal to c. They are laws for all numbers, not just the booleans.

The arithmetic expression x - y varies directly with x and inversely with y. Thus if we increase x, we increase x - y, and if we decrease y we increase x - y. We calculate:

x - y increase x to x+ 1 and so increase the whole expression
≤ (x+ 1) - y decrease y to y-1 and so increase the whole expression ≤ (x+l)- (y-l)

Similarly the boolean expression x ≥ y varies directly with x and inversely with y (no matter whether x and y are numbers and ≥ is number comparison, or x and y are boolean and ≥ is reverse implication, or x and y are a mixture of number and boolean). We calculate as follows:

x≥y increase x to x+l and so increase the whole expression
≤ (x+ 1) ≥ y decrease y to y-1 and so increase the whole expression ≤ (x+l) ≥ (y-l)

It is exactly the same calculation. By unifying number algebra with boolean algebra we carry our ability to calculate over from numbers to booleans.

Unified Development

Suppose we start with boolean algebra in the unified notation, with the terminology "top", "bottom", "minimum", "maximum", "less than", and so on. Now we say: what a pity that x= -x has no solution; let's give it one. The new solution is denoted 0. While gaining a solution to some boolean expressions, we lose some laws such as the law of the excluded middle x ∨ -x.

Now we have an algebra of three values: a, b, 0. In one application they can be used to represent "yes", "no", and "maybe"; in another they can be used to represent "large", "small", and "medium". This algebra has 27 one-operand operators, one of which is -, defined as

x	⊤	0	⊥
−x	⊥	0	⊤

In has 19683 two-operand operators, four of which are:

xy	⊤⊤	⊤0	⊤⊥	0⊤	00	0⊥	⊥⊤	⊥0	⊥⊥
x=y	⊤	⊥	⊥	⊥	⊤	⊥	⊥	⊥	⊤
x≤y	⊤	⊥	⊥	⊤	⊤	⊥	⊤	⊤	⊤
x≦y	⊤	0	⊥	⊤	0	0	0	0	0
x⊕y	⊥	⊤	0	⊤	0	⊥	0	⊥	⊤

Whether ≤ or ? or another operator represents implication in the presence of uncertainty can be debated, but the algebra is not affected by the debate. The operator ⊕ is modular (or circular) addition, and the other operators of modular arithmetic can be given similarly.

We might continue our development with a four-valued algebra and five-valued algebra, but at this point I recommend filling in the space between a and 0, and between 0 and b, with all the integers. And then on to the rationals, the reals, and the complex numbers as usual.

Quantifiers

There are several notations that introduce a local (bound, dummy) variable. For example,

$$\sum_{x=0}^{\infty} fx \quad \int_a^b fx\,dx \quad \forall x : D \cdot Px \quad \{fx \mid x \in D\}$$

The introduction of the local variable and its domain are exactly the job of the function notation, so all expressions requiring a local variable can be uniformly expressed as an operator applied to a function. If the body of a function is a number expression, then we can apply + to obtain the sum of the function results. For example,

$$+\langle n: \text{nat} \to 1/2^n \rangle$$

There is no syntactic ambiguity caused by this use of +, so no need to employ another symbol ∑ for addition. We can apply any associative symmetric operator, such as

$$\times\langle n: \text{nat} \to 1/2^n \}$$
$$\wedge\{n: \text{nat} \to n>5\}$$
$$\vee\,(n: \text{nat} \to n>5)$$

The minimum operator ∧ replaces "for all", and the maximum operator ∨ replaces "there exists". By applying = and ≠ to functions we obtain the two independent parity operators. Set comprehension and integrals can be treated this same way.

If function f has domain D, then f=⟨x: D →fx⟩, so quantifications traditionally written

$$\sum_{xd} fx \quad \forall x : D \cdot Px$$

which we have just learned to write as

$$+\langle x: D \to fx\rangle \wedge\langle x: D \to Px\rangle$$

can be written even more succinctly as

$$+f \ \wedge P$$

Using juxtaposition for composition, deMorgan's laws

$$\neg(\forall x: D \cdot Px) \equiv (\exists x: D \cdot \neg Px) \quad \neg(\exists x: D \cdot Px) \equiv (\forall x: D \cdot \neg Px)$$

become

$$-\wedge P = \vee -P \quad -\vee P = \wedge -P$$

or even more succinctly

$$(-\wedge) = (\vee -) \quad (-\vee) = (\wedge -)$$

The Specialization and Generalization laws say that if y is an element of D,

$$(\forall x: D \cdot Px) \Rightarrow Py \qquad Py \Rightarrow (\exists x: D \cdot Px)$$

They now become

$$\wedge p \le py \qquad Py \le \vee P$$

which say that the minimum item is less than or equal to any item, and any item is less than or equal to the maximum item. These laws hold for all numbers, not just for the booleans.

Given function f, all function values fx are at least y if and only if the minimum function value fx is at least y. Traditionally, that's a universal quantification equated to a minimum. In unified algebra, it is just factoring. Leaving the non-null domain of f implicit, we write

$$\wedge\langle x \to fx \ge y\rangle \text{ factor out } \ge y$$
$$= \ \wedge\langle x \to fx\rangle \ge y$$
$$= \ \wedge f \ge y$$

If we go in the other direction, "unfactoring" is called "distribution". And it works whether fx and y are numbers and ≥ is the number ordering, or fx and y are booleans and ≥ is reverse implication. It's no different from the factoring/distribution law that says the minimum value of (fx – y) equals (the minimum value of (fx) – y.

$$\wedge\langle x \to fx - y\rangle \text{ factor out } -y$$
$$= \ \wedge\langle x \to fx\rangle - y$$
$$= \ \wedge f - y$$

If we factor from the other side of the - sign, we have to change minimum to maximum:

$$\wedge\langle x \to y - fx\rangle \text{ factor out } y-$$
$$= \ y - \vee\langle x \to fx\rangle$$
$$= \ y - \vee f \text{ And similarly}$$
$$\wedge\langle x \to y \ge fx\rangle \text{ factor out } y\ge$$
$$= \ y \ge \vee\langle x \to fx\rangle$$
$$= \ y \ge \vee f$$

Once again, it works for numbers and booleans equally well. Unified algebra gives us many other factoring/ distribution laws just like these.

The goal is to create an algebra that's easy to learn and easy to use. That goal is not always consistent with traditional mathematical terminology and symbology. Readers are cautioned against matching the algebra directly with their own familiar terms and symbols. Although I have been using the words "minimum" and "maximum" for ∧ and ∨,

the words "greatest lower bound" and "least upper bound", or "infimum" and "supremum", may be more traditional in some contexts. For example,

$$\wedge(n: \text{nat} \rightarrow 1/n) = 0$$

Even more caution must be used with the words "all" and "exists". Intuition about existence in mathematics (like intuition about anything else) depends on what you have learned. We tend to believe that what we have learned is true. But mathematical truth is constructed, and we must be open to the possibility of constructing it differently. Unlearning can be more difficult than learning.

Quantifier examples

Is $(\exists x \cdot Px) \Rightarrow (\forall y \cdot Qy)$ equivalent to $\forall x \cdot \forall y \cdot (Px \Rightarrow Qy)$? Even experienced logicians don't find it obvious. To see whether they are equivalent, those who reason informally say things like "suppose some x has property P ", and "suppose all y have property Q ". They are led into case analyses by treating $\forall$ and $\exists$ as abbreviations for "for all" and "there exists" (as they originally were). Of the very few who reason formally, most don't know many laws; perhaps they start by getting rid of the implications in favor of negation and disjunction, then use deMorgan's laws. Let me rewrite the questionable equivalences in the new notations.

$$(\vee P \leq AQ) = \wedge\langle x \rightarrow \wedge\langle y \rightarrow Px \leq Qy\rangle\rangle$$

We might read the left side as saying that the maximum P is less than or equal to the minimum Q, and we might read the right side as saying that all P are less than or equal to all Q. Informal readings can be misleading, and we should never attach our understanding to an informal reading, but sometimes we can get inspiration from it. In this case, the reading sounds reasonable enough to suggest we might prove it, and not just for booleans, but for all numbers. Leaving the non-null domains implicit, here's the proof:

$$\begin{aligned} & \wedge\langle x \rightarrow \wedge\langle y \rightarrow Px \leq Qy\rangle\rangle \\ & \quad \text{factor out } Px \leq \\ = \quad & \wedge\langle x \rightarrow Px \leq \wedge Q\} \\ & \quad \text{factor out } \leq \wedge Q \\ = \quad & \vee P \leq \wedge Q \end{aligned}$$

Let L be a nonempty list (a function whose domain is an initial segment of the naturals). +L is its sum, and $\vee$L is its maximum; let #L be its length. We can say that the average item in the list is less than or equal to the maximum item as follows.

$$\begin{aligned} & +L/\#L \leq \vee L \\ & \text{now apply } > 1 \text{ to both sides of the inequality} \\ \leq \quad & (+L/\#L > 1) \leq (\vee L > 1) \\ & \text{multiply by \#L; distribute } >1 \\ = \quad & (+L > \#L) \leq \vee\langle i \rightarrow Li > 1\rangle \end{aligned}$$

leaving the domain implicit. The bottom line is the "pigeon-hole principle"; it says that if the total number of things is greater than the number of places to put them, then some place has more than one thing in it. Notice what has happened: we read $\vee$ as "maximum" on the top line, and as "some" on the bottom line; we read $\leq$ as "less than or equal to" on the top line, and as "if then" on the bottom line.

Here is a further illustration of the benefits of unified algebra. Let f be a function from the naturals to the reals. If f is non-decreasing, then f0 is its minimum. Traditionally, this might be written (leaving the domain implicit) as

$$(\forall n \cdot fn \leq f(n+l)) \Rightarrow (f0 = \text{MIN}\{\{0 \leq n < \infty\})$$

Rewriting this in the new notation, and weakening it to say that f0 is less than or equal to the minimum, we get

$$\wedge\langle n \rightarrow fn \leq f(n+l)\rangle \leq (f0 \leq \wedge f)$$

Now we apply the portation law, which says that for boolean a and any b and c,

$$(a \leq (b \leq c)) = (a \wedge b \leq c)$$

to obtain

$$f0 \wedge \wedge\langle n \rightarrow fn \leq f(n+l)\rangle \wedge f$$

If f happens to have a boolean range, this is induction, more traditionally written

$$f0 \wedge (\forall n \cdot fn \Rightarrow f(n+l)) \Rightarrow (\forall n \cdot fn)$$

Thus we see induction as a special case of a more general law saying that the first item in a non-decreasing sequence is its minimum.

The seminal work by Boole on boolean algebra refers to both logic and probability. The standard theory of probability assigns 0 to an event that cannot happen, 1/2 to an event that is equally likely to happen or not happen, and 1 to an event that is certain to happen. In a set of events in which exactly one event must happen, the probabilities sum to 1. The integral of a probability distribution must be 1.

Perhaps there is another way to develop probability theory based on unified algebra. Perhaps an event that cannot happen has probability $\perp$, an event that is equally likely to happen or not happen has probability 0, and an event that is certain to happen has probability a. In a set of events in which exactly one event must happen, the average probability is 0. The integral of a probability distribution must be 0. Perhaps the new probability space is related to the logarithm of the old space; essentially, probabilities are replaced by information content. My hope is that the complicated formulas for distributions in the standard theory can be simplified by transforming the space of probabilities.

Metalogic

In the study of logic, at or near the beginning, logicians present the symbol I- to represent theoremhood. I ask you to put yourself in the place of a beginning student. This symbol is applied to a boolean expression just like the boolean operators; but we know all the boolean operators and this isn't one of them. To make things worse, there are different levels of meta-operators. Proof rules are sometimes presented using a horizontal bar, which is yet another level of implication. Consider, for example, the Modus Ponens proof rule, which uses all three kinds of implication:

$$\frac{A \succ x, B \succ x \Rightarrow y}{A, B \succ y}$$

Rewriting comma as conjunction, and turnstile and bar as implication, we get a tautology:

$$(A \Rightarrow x) \wedge (B \Rightarrow (x \Rightarrow y)) \Rightarrow (A \wedge B \Rightarrow y)$$

We have a name, "theorem", for a boolean expression that can be simplified to a, and an operator, „, whose purpose is to identify theorems. Strangely, logicians have not introduced a name, say "antitheorem", for a boolean

expression that can be simplified to ⊥, and no operator such as …, whose purpose is to identify antitheorems. Perhaps that's because "antitheorem" just means "negation of a theorem" in those logics having negation and an appropriate proof rule. But we bother to name both booleans, even though one is just the negation of the other.

I propose that logicians can improve metalogic by taking another lesson from programming. Instead of „ and …, we need only one operator to serve both purposes. It is called an interpreter. I want Is to be a theorem if and only if s represents a theorem, and an antitheorem if and only if s represents an antitheorem. It is related to „ and … by the two implications

$$s \leq Is \leq - \ldots s$$

In fact, if we have defined „ and …, those implications define I. But I want T to replace „ and …, so I shall instead define it by showing how it applies to every form of boolean expression. Here is the beginning of its definition.

$$I"\ a" = a$$
$$I"\bot = \bot$$
$$I\ ("-"\ s) = -Is$$
$$I(s"\wedge"t) = Is \wedge It$$
$$I(s"\vee"t) = Is \vee It$$

And so on. In a vague sense Z acts as the inverse of quotation marks; it "unquotes" its operand. That is what an interpreter does: it turns passive data into active program. It is a familiar fact to programmers that we can write an interpreter for a language in that same language, and that is just what we are doing here. Interpreting (unquoting) is exactly what logicians call Tarskian semantics. In summary, an interpreter is a better version of „ and strings make metalevel operators unnecessary.

Using I, the famous Godel incompleteness proof is just 3 lines. Suppose that every boolean expression is either a theorem or an antitheorem (a complete logic), and define Q by

$$Q = "-IQ"$$

Then

$$IQ \quad \text{replace Q with its equal}$$
$$= \quad I"-IQ" \quad \text{I unquotes}$$
$$= \quad -IQ$$

Conclusions

This paper has not presented a detailed proposal for a change to our primary and secondary mathematics curriculum, but it has presented the case for making a change, and several suggestions. The main suggestion is to unify boolean algebra with number algebra so that we can begin with the simplest algebra and move smoothly to the more complicated algebras, all using the same notations and in the same calculational framework. To use logic well, one must learn it early, and practice a lot. Fancy versions of logic, such as modal logic and metalogic, can be left to university study, but there is a simple basic algebra that can be taught early and used widely.

References

1. Gries D. Improving the Curriculum through the teaching of Calculation and Discrimination. Communications of the ACM 1991;34(3):45-55.
2. Gries D, Schneider FB. A Logical Approach to Discrete Math, Springer-Verlag 1993.
3. Herstein IN. Topics in Algebra, Blaisdell 1964, 323.
4. Katz RH. Contemporary Logic Design, Benjamin Cummings 1994, 10.
5. Hehner ECR. "Beautifying G5del", chapter 18 in Beauty is our Business, a birthday tribute to Edsger Dijkstra, Springer-Verlag 1990.
6. Hehner ECR. "Unified Algebra", www.cs.toronto.edu/-hehner/UA.pdf.
7. Boute R. Binary Algebra and Functional Predicate Calculus: a Practical Approach, University of Ghent, Belgium 1999.
8. Church A. The Calculi of Lambda-Conversion. Annals of Mathematical Studies, Princeton University Press 1941, 6.
9. Dijkstra EW, Scholten CS. Predicate Calculus and Program Semantics, Springer-Verlag 1990.
10. Allen LE. Symbolic Logic: a Razor-Edged Tool for Drafting and Interpreting Legal Documents. Yale Law Journal 1957;66:833-879.
11. Bochenski JM. A History of Formal Logic second edition, translated and edited by Ivo Thomas, Chelsea Publishing, New York 1970.