29 (bracket algebra) Here is a new way to write binary expressions. An expression can be empty; in other words, nothing is already an expression. If you put a pair of brackets around an expression, you get another expression. If you put two expressions next to each other, you get another expression. For example,

( )(( ))((( ))( ))

is an expression. The empty expression is bracket algebra's way of writing $\top$; putting brackets around an expression is bracket algebra's way of negating it, and putting expressions next to each other is bracket algebra's way of conjoining them. So the example expression is bracket algebra's way of saying

$\neg\top\wedge\neg\neg\top\wedge\neg(\neg\neg\top\wedge\neg\top)$

We can also have variables anywhere in a bracket expression. There are three rules of bracket algebra. If $x$, $y$, and $z$ are any bracket expressions, then

| | | | |
|---|---|---|---|
| $((x))$ | can replace or be replaced by | $x$ | double negation rule |
| $x( )y$ | can replace or be replaced by | $( )$ | base rule |
| $x\,y\,z$ | can replace or be replaced by | $x'\,y\,z'$ | context rule |

where $x'$ is $x$ with occurrences of $y$ added or deleted, and similarly $z'$ is $z$ with occurrences of $y$ added or deleted. The context rule does not say how many occurrences of $y$ are added or deleted; it could be any number from none to all of them. To prove, you just follow the rules until the expression disappears. For example,

| | | |
|---|---|---|
| | $((a)b((a)b))$ | context rule: empty for $x$, $(a)b$ for $y$, $((a)b)$ for $z$ |
| becomes | $((a)b( ))$ | base rule: $(a)b$ for $x$ and empty for $y$ |
| becomes | $( ( ))$ | double negation rule |
| becomes | | |

Since the last expression is empty, all the expressions are proven.

(a) Rewrite the binary expression

$\neg(\neg(a\wedge b)\wedge\neg(\neg a\wedge b)\wedge\neg(a\wedge\neg b)\wedge\neg(\neg a\wedge\neg b))$

as a bracket expression, and then prove it following the rules of bracket algebra.

(b) As directly as possible, rewrite the binary expression

$(\neg a\Rightarrow\neg b)\wedge(a\neq b)\vee(a\wedge c\Rightarrow b\wedge c)$

as a bracket expression, and then prove it following the rules of bracket algebra.

(c) Can all binary expressions be rewritten reasonably directly as bracket expressions?

(d) Can $x\,y$ become $y\,x$ using the rules of bracket algebra?

(e) Can all theorems of Binary Theory, rewritten reasonably directly as bracket expressions, be proven using the rules of bracket algebra?

(f) We interpret empty as $\top$, brackets as negation, and adjacency as conjunction. Is there any other consistent way to interpret the symbols and rules of bracket algebra?

After trying the question, scroll down to the solution.

(a) Rewrite the binary expression

$$\neg(\neg(a\wedge b)\wedge\neg(\neg a\wedge b)\wedge\neg(a\wedge\neg b)\wedge\neg(\neg a\wedge\neg b))$$

as a bracket expression, and then prove it following the rules of bracket algebra.

§

| | |
|---|---|
| $((ab)((a\ )b)(a(b))((a\ \ )(b)))$ | context: insert $b$ |
| $((ab)((ab)b)(a(b))((a\ \ )(b)))$ | context: delete $(ab)$ |
| $((ab)(\ \ \ b)(a(b))((a\ \ )(b)))$ | context: insert $(b)$ |
| $((ab)(\ \ \ b)(a(b))((a(b))(b)))$ | context: delete $(a(b))$ |
| $((ab)(\ \ \ b)(a(b))(\ \ \ (b)))$ | context: delete $(b)$ twice |
| $((ab)(\ \ \ b)(a\ \ )(\ \ \ \ ))$ | base |
| $(\ \ \ \ \ \ \ \ \ \ (\ \ \ \ ))$ | double negation |
| | this line is empty |

(b) As directly as possible, rewrite the binary expression

$$(\neg a\Rightarrow\neg b)\wedge(a\neq b)\ \vee\ (a\wedge c\Rightarrow b\wedge c)$$

as a bracket expression, and then prove it following the rules of bracket algebra.

§ The operators $\Rightarrow$, $\neq$, and $\vee$ are not in bracket algebra, so I'll replace $x\Rightarrow y$ with $\neg(x\wedge\neg y)$, $x\neq y$ with $\neg(x\wedge y)\wedge\neg(\neg x\wedge\neg y)$, and $x\vee y$ with $\neg(\neg x\wedge\neg y)$.

| | |
|---|---|
| $((((a))((b)))(ab)((a)(b)))((ac(bc))))$ | double negative twice |
| $((((a)\ \ b\ )(ab)((a)(b)))\ \ ac(bc)\ \ )$ | context: final $a$ deletes earlier $a$'s |
| $((((\ )\ \ b\ )(\ b)((\ )(b)))\ \ ac(bc)\ \ )$ | base twice |
| $((((\ )\ \ \ \ )(\ b)((\ )\ \ ))\ \ ac(bc)\ \ )$ | double negative twice |
| $((\ \ \ \ \ \ (\ b)\ \ \ \ \ )\ \ ac(bc)\ \ )$ | double negative |
| $(\ \ \ \ \ \ \ \ b\ \ \ \ \ \ \ \ ac(bc)\ \ )$ | context: first $b$ deletes other one |
| $(\ \ \ \ \ \ \ \ b\ \ \ \ \ \ \ \ ac(\ c)\ \ )$ | context: first $c$ deletes other one |
| $(\ \ \ \ \ \ \ \ b\ \ \ \ \ \ \ \ ac(\ \ )\ \ )$ | base rule |
| $(\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ (\ \ )\ \ )$ | double negation |
| | this line is empty |

(c) Can all binary expressions be rewritten reasonably directly as bracket expressions?

§ In a sense, no, and in another sense, yes. The operators $\vee$, $\Rightarrow$, $\Leftarrow$, $=$, and $\neq$ are not in bracket algebra, so no, expressions using those operators cannot be written as bracket algebra expressions. But for any binary expression there is an equivalent bracket algebra expression because negation and conjunction are a basis for all binary operators. So in that sense, yes, all binary expressions can be rewritten as bracket expressions.

(d) Can $x\,y$ become $y\,x$ using the rules of bracket algebra?

§ Yes, I'll use $a$ and $b$ so I can show the use of the rules more clearly.

| | |
|---|---|
| $a\,b$ | context: empty for $x$, $a$ for $y$, $b$ for $z$ |
| $a\,b\,a$ | context: $ab$ for $x$, $a$ for $y$, empty for $z$ |
| $b\,a$ | |

(e) Can all theorems of Binary Theory, rewritten reasonably directly as bracket expressions, be proven using the rules of bracket algebra?

§ Yes. We only need to consider a binary theorem $X$ formed using $\neg$, $\wedge$, $\top$, and variables. Theoremhood of such expressions is given by the Completion Rule, so we only need to simulate this one rule in bracket algebra. In part (a) we proved the equivalent of

$$a\wedge b\ \vee\ \neg a\wedge b\ \vee\ a\wedge\neg b\ \vee\ \neg a\wedge\neg b$$

which says that variables $a$ and $b$ must have one of their 4 possible assignments of values. This can be generalized to any number of variables, but we will use just two variables for illustration.

$X$ double negation
$((X))$ now follow the steps of part (a) backwards to insert it inside the final bracket
$((X)((ab)((a)b)(a(b))((a)(b))))$ context: insert $(X)$ into each of the assignments
There are $4$ assignments here, but in general, $n$ variables give us $2^n$ assignments
$((X)((ab(X))((a)b(X))(a(b)(X))((a)(b)(X))))$ In the context of each assignment,
we can replace each appearance of a variable in $X$ by either empty or ( ),
then simplify the resulting expression without variables by repeatedly
applying base and double-negation rules. This in effect evaluates $X$.
Since $X$ is a binary theorem, it simplifies to empty.
$((X)((ab( ))((a)b( ))(a(b)( ))((a)(b)( ))))$ base $2^n$ times
$((X)((( ))(( ))(( ))(( ))))$ double negation $2^n$ times
$((X)( ))$ base
$(( ))$ double negation
this line is empty

(f) We interpret empty as $\top$, brackets as negation, and adjacency as conjunction. Is there any other consistent way to interpret the symbols and rules of bracket algebra?

§ Yes. Interpret empty as $\bot$, brackets as negation, and adjacency as disjunction, and proof means following the rules until the expression becomes $( )$.

For more on bracket algebra, see Boundary Algebra.