

# What are the rules of “elementary algebra”

James Davenport & Chris Sangwin

Universities of Bath & Birmingham

7 July 2010

# Setting

- ▶ A relatively traditional mathematics course, at, say first-year undergraduate level.
- ▶ **But** Computer-Aided Assessment is in use.
- ▶ One such example is WeBWork, another is MapleTA.
- ▶ “Harness the power of technology to improve teaching and learning” [AMS Notices, June 2009].

## Web-based Assessment and Testing Systems

“Homework software has the potential to handle the grading of homework at a low cost. While this software has the limitation of requiring a concise answer — an algebraic expression or a multiple-choice response — it also has an important advantage over hand grading. Namely, if a student’s answer to a problem is wrong, the student learns of the mistake immediately and can be allowed to try the problem or a similar problem repeatedly until the right answer is obtained.” [AMS]

## Automatically generated problems

Means automatically generated answers *and marking schemes*.

- ▶ Parsing the student's answer (non-trivial — see next slide)
- ▶ Is the student's answer mathematically correct?
- ▶ Is the student's answer pedagogically correct?
- ▶ So what mark does it get (assuming we are doing more than true/false marking)?

Marking other than true/false was not discussed by the AMS, but seems important to us.

# Typical computer aided assessment

What is

$$\frac{d \sin^2 2x}{dx}?$$

4sin(2x)\*cos(2x)

Your last answer was interpreted as:

$$4 \cdot \sin(2 \cdot x) \cdot \cos(2 \cdot x)$$

Correct answer, well done.


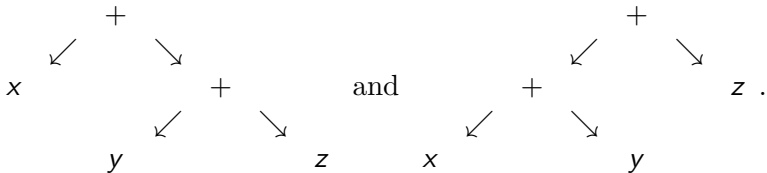
Your mark for this attempt is 1. 

Figure: STACK system [?]

## Phase 1: Addition and Multiplication only

This is (close to) AC-rewriting. We can distinguish levels:

1. Textual identity. Here  $x+ y$  and  $x+y$  are different (space). Similarly  $x+10$  and  $x+010$  are different (leading zero).
2. Equality after lexical analysis. Above examples are the same, but not  $x+y*z$  versus  $x+(y*z)$ .
3. Equality as binary parse trees. This would consider  $x+y*z$  and  $x+(y*z)$  as the same. However, this interpretation would distinguish  $x+(y+z)$  from  $(x+y)+z$ , being respectively the trees



Being based on binary parse trees, it requires an operator precedence rule (is  $+$  left-associative or right-associative) to decide which of these becomes the representation of  $x+y+z$ .

## Phase 1: Addition and Multiplication (continued)

1. Textual identity.
2. Equality after lexical analysis.
3. Equality as binary parse trees. However, this interpretation would distinguish  $x+(y+z)$  from  $(x+y)+z$ .
4. We can avoid this if we allow  $n$ -ary trees, parsing  $x+y+z$  as

$$\begin{array}{ccc} & + & \\ & \swarrow \downarrow \searrow & \\ x & & y & & z \end{array}, \quad (1)$$

but this is now a third tree, different from the two above.

5. Solved if we *flatten* the associative operators  $+$  and  $*$ , which would transform both trees into the tree in (1).
6. But “the rules of elementary algebra” include the facts that  $+$  and  $\times$  are commutative, as well as associative. So compare the children of a  $+$  (or  $*$ ) node as *(multi)sets*, rather than as lists, so now the tree in (1) has the same children, admittedly in a different order, as the trees for  $y+(z+x)$  etc.





## So far, so good

Depending on the precise definition of “obviously equal”, somewhere between levels 6 and 9 lies a system that gets many examples “right”, in the sense that the user is not upset that the system refuses to recognise as equal expressions in  $+$  and  $*$  that are “obviously equal”.

However, the grammar of even elementary algebra is larger than this, and includes  $-$  (in both unary and binary forms) and  $/$ .

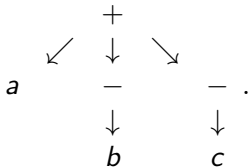
Practically any system for processing mathematics has to deal with these, and their impact on the problem of “equality modulo the usual rules of elementary algebra”.

It might be thought that  $/$  and (binary)  $-$  posed the same problems, being the inverse operations to  $*$  and  $+$  respectively, but in fact the unary operators behave differently.

## Subtraction

Here we have both unary and binary  $-$ . We seem to make the transformation  $a-b \rightarrow a+(-b)$  at a very early stage in our mental processing.

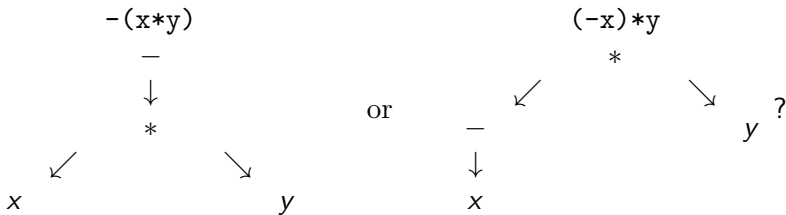
Hence the expression  $a-b-c$  is thought of as  $a+(-b)+(-c)$ , and, in levels 4 and beyond, becomes the tree



Level 6 would then regard this as equal to the tree from  $a-c-b$ . It would, however, also regard this as equal to the expression  $-b+a-c$ . Logically, one cannot object to this, but nonetheless we “tend to prefer” one of the other forms. Convention urges minimality here, since the other forms require one less symbol.

# Negation in Parse Trees

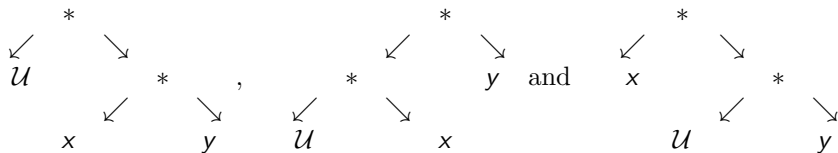
Does  $-x*y$  parse as



These are "obviously equal".

## Our solution

In STACK we replace “unary minus” by multiplication by a special token, hereafter  $\mathcal{U}$ . Hence  $-(x*y)$ ,  $(-x)*y$  and  $x*(-y)$  become the trees

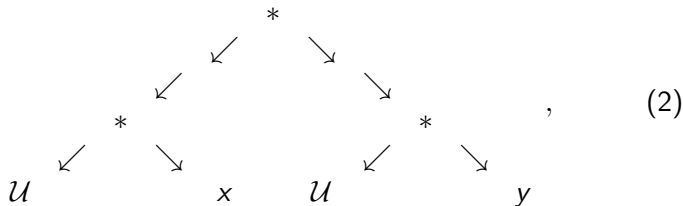


respectively.

In the presence of flattening and commutativity of  $*$ , these trees are regarded as equal.

## What about $(-x)*(-y)$ ?

This corresponds to the tree

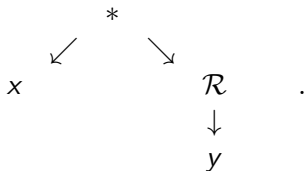


which is equivalent to the tree from  $-(-x*y)$  and many other variants with (multi)set of children  $\{U, U, x, y\}$ , but *not* equivalent to the tree from  $x*y$  until we get to level 7.

Algebraically  $U$  behaves like  $-1$ , but is distinguished from it at this level so as not to be confused with an explicit  $-1$  entered by the user, i.e.  $-x*y$  versus  $x*(-1)*y$ .

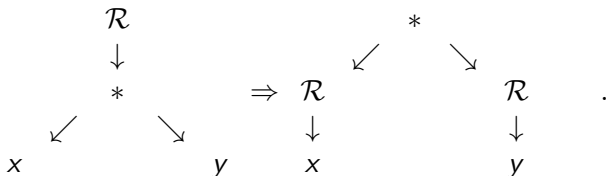
## Division

We use “reduction to the unary case”, even though the surface representation is different. We consider a *unary* reciprocal operator  $\mathcal{R}$ , and regard  $x/y$  as generating the tree



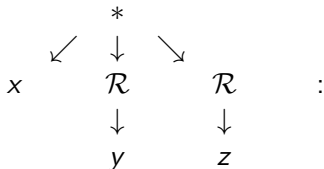
We need one further rule

*“ $\mathcal{R}$  distributes over multiplication”, i.e.*

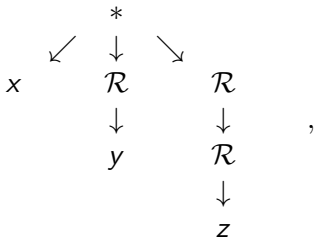


## Division continued

$(x/y)/z$ ,  $x/y/z$  and  $x/(y*z)$  all generate



$x/(y/z)$  generates



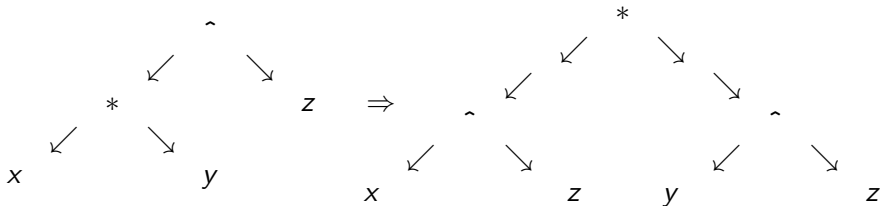
Doesn't simplify at level 6, but  $\mathcal{R}(\mathcal{R}(z)) \Rightarrow z$  should be readily available in the toolkit of an exercise writer (e.g. at level 7).

## (Integer) Powers

The first challenge is that the distributive law for exponentiation over multiplication

$$(ab)^c = a^c b^c$$

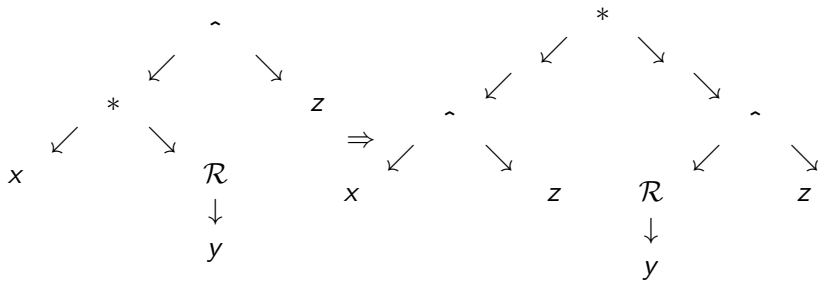
seems different from the distributive law for multiplication over addition: we do not regard  $(xy)^2$  as better or worse than  $x^2y^2$ . This is easily implemented as a tree transformation:





## (Integer) Powers continued

We do not regard  $\left(\frac{x}{y}\right)^2$  as better or worse than  $\frac{x^2}{y^2}$ . The previous transformation automatically leads to

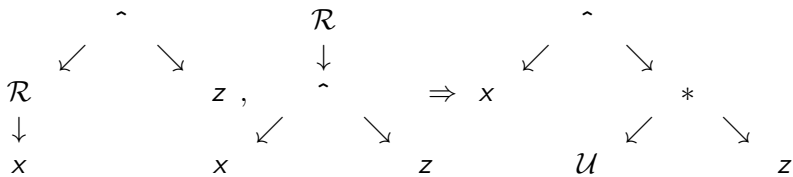


but this is not quite what was desired.

It is tempting to try to “fix” this by rules taking  $\mathcal{R}(x)^y$  to  $\mathcal{R}(x^y)$ .

## (Integer) Powers continued

However, we accept that  $\mathcal{R}$  is really raising to the power  $-1$ , and introduce the rules



$1/x^{(-2)}$  is then represented as  $x^{U*U*2}$ .

Author choice whether to regard  $1/x^{(-2)}$  as a clumsy expression which does not deserve to be simplified, or work at level 7, in which case  $U * U * z \Rightarrow z$  (or the pair  $U * U \Rightarrow 1$  and  $1 * z \Rightarrow z$ ) need to be added to the ruleset.

# Conclusion

- ▶ Straight AC rewriting of  $+$ ,  $*$  is relatively easy
- ▶ Adding  $-$  and  $/$  is not easy

But the “unary operator” technique seems promising

- ▶ Integer powers aren't that easy, but doable
- ▶ There are good mathematical reasons why using non-integer powers is harder!