

# Using Term Rewriting Systems in Algebra

Jeff Clark  
Elon University  
<http://frodo.elon.edu>

January 8, 2004

Many algebraic systems have natural equivalence relations defined on them, where equivalent things are not equally simple to work with.

- $1/2$  and  $352/704$  are equivalent representations of the same fraction, but the first form is much easier to work with.
- $x + x$  and  $2x$  are equivalent polynomials, but the second form is easier to work with.

- 

$$\{(1, 2, 3)^T, (4, 5, 6)^T, (7, 8, 8)^T\}$$

and

$$\{(1, 0, 0)^T, (0, 1, 0)^T, (1, 0, 0)^T\}$$

are both bases for the same vector space over  $\mathbb{R}$ , but the second is much easier to work with.

For many such systems, it is possible to identify a partial order “is simpler than” that is consistent with the equivalence relationship. If so, then we can take any object within the system and try to replace it by a simpler equivalent object.

This is what we do when we simplify fractions, put polynomials into standard form, or change to simpler bases in a vector space.

Problems:

- How do we find a simpler object?
- If we keep simplifying, will we ever find a simplest equivalent object?
- If we do find a simplest equivalent object, is it unique?

Equivalence relations for mathematical systems are often generated by a finite list of axioms, each claiming that two expressions are equivalent. For example, a group has as axioms the associative law and the definitions of the identity and inverses. A commutative group has an additional axiom involving commutativity, etc.

Given such a finite list of axioms, if we can consistently designate one form as simpler than the other, we can use the axiom as a rule for simplifying expressions.

Example: group theory. For simplicity of working with the algorithm we will use prefix notation. Let  $m$  represent the multiplication operator,  $i$  represent the inversion operator, and  $e$  represent the constant identity. Let  $a, b, c$  be variable names. Then it is theoretically possible to define a group using associativity, a left-identity axiom, and a left-inverse axiom.

- $mmabc = mambc$

- $mea = a$

- $miaa = e$

It is possible to define a partial order on this system using two properties:

- If  $x$  is a term containing a variable  $y$ , then  $y$  is simpler than  $x$ .
- Given  $fx_1x_2\dots$  and  $fy_1y_2\dots$  we can compare them lexicographically.

We can replace each axiom by a rewrite rule:

- $mmabc \rightarrow mambc$

- $mea \rightarrow a$

- $miaa \rightarrow e$

In this fashion, we can solve the first problem: given a finite list of axioms, we can try to find a partial order that is consistent with the equivalence relation that they define. Then we can transform our list of axioms into a list of rewrite rules. Given a term to simplify, we keep on applying the rewrite rules, always finding something simpler.

For example:

$$\begin{aligned}mmiaaia &\rightarrow meia \\ &\rightarrow ia\end{aligned}$$

Fortunately, we can show that this set of rewrite rules will always terminate: the number of symbols in each term will either decrease, or they will stay the same and some  $m$ 's will move to the right. Neither of these options can be repeated indefinitely.

Unfortunately, this result is not unique. If we apply the first rewrite rule:

$$mmiaaia \rightarrow miamaia$$

which can not be simplified any further.

This ambiguity arises because of overlap between the associative and inverse axioms. One way of dealing with this sort of ambiguity is to search out all possible overlaps (which can be done immediately with any two specific axioms) and add them as new axioms.

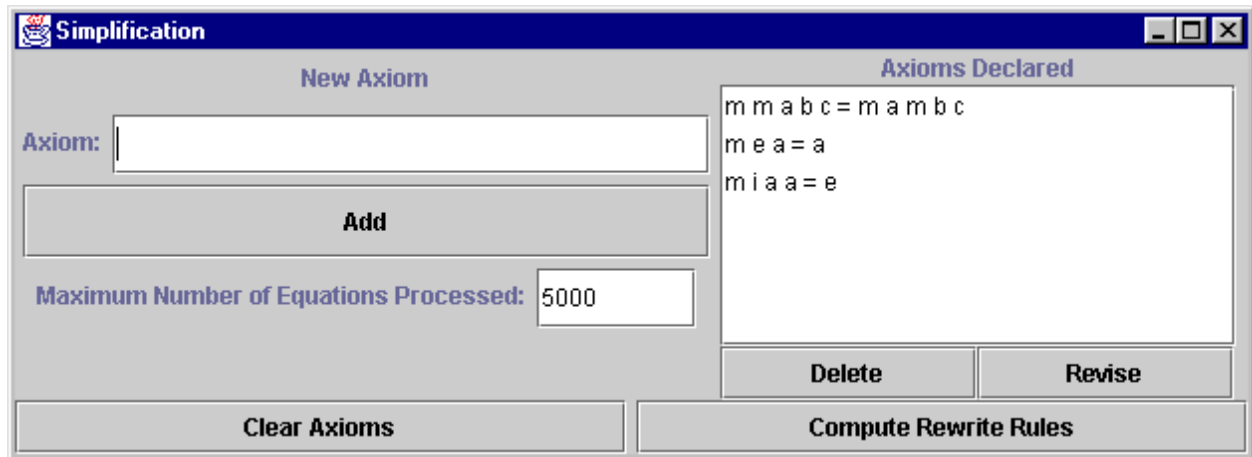
If this process converges to yield a finite set of rules, we are done. It need not converge.

If it does converge, the system of rewrite rules can be used for automated theorem-proving.

Knuth and Bendix in their original paper specified a way of defining a partial order based on weights given to the operators as well as the order in which they are listed.

*Simplification* is a program that I have written implementing the Knuth-Bendix procedure.

Simplification		Operators Declared		New Variable		Variables Declared	
<b>Name:</b>	<input type="text"/>	e: degree = 0, weight = 1 m: degree = 2, weight = 0 i: degree = 1, weight = 0		<b>Name:</b>	<input type="text"/>	a b c	
<b>Degree:</b>	<input type="text"/>			<b>Add</b>			
<b>Weight:</b>	<input type="text" value="1"/>			<b>Delete</b>	<b>Revise</b>		
<b>Add</b>		<b>Delete</b>		<b>Delete</b>		<b>Revise</b>	
<b>Clear All Declarations</b>				<b>Enter Axioms</b>			





```
Simplification
Rewrite Rules
#1: m e a -> a
#2: m i a a -> e
#3: m m a b c -> m a m b c
#4: m i a m a b -> b
#8: i e -> e
#10: i i a -> a
#11: m a e -> a
#12: m a i a -> e
#13: m a m i a b -> b
#16: i m a b -> m i b i a
```

Note that among the results are the consequences that the inverse also is a right-inverse and that the identity is also a right-identity. Checking the log file generated by *Simplification* provides a derivation of these results.

*Simplification* is written in *Python*, an object-oriented language known for its clarity and concision. *Python* programs can run on their own, or they can be compiled into *Java* bytecode and/or JAR files. *Simplification* itself is a JAR file.

*Simplification* can be used to explore axiom systems and see what their consequences are. I hope that it will prove useful to students and faculty in upper-level abstract algebra classes.

## References

This talk and the related software can be found at <http://frodo.elon.edu> under the link “Simplification” .

1. Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
2. Donald E. Knuth and P.B. Bendix. Simple word problems in universal algebra. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.
3. Enno Ohlebusch. *Advanced Topics in Term Rewriting*. Springer-Verlag, 2002.

4. All things *Python* can be found starting at <http://www.python.org>.
  
5. All things *Jython* (using *Python* to create *Java* bytecode) can be found starting at <http://www.jython.org>.