Colloquium: Solving word equations

Speaker: Dr Murray Elder, The University of Newcastle

Abstract: Consider this pattern-matching problem: given two finite strings of letters $a, b, c, \ldots$ and *variables* $X, Y, Z, \ldots$, the goal is to substitute the variables for finite strings of letters so that the two expressions are identical. For example,

$$aX \stackrel{?}{=} Xa$$

Replacing X by a makes the two sides the same, aa. In fact replacing X by any nonnegative number of a's is okay (including 0). Replacing X by some word with a b in it, is not (prove it). So we claim the set of all solutions to this *word equation* is $X = a^i, i \in \mathbb{N}$.

Here is another example:

$$aXXb \stackrel{?}{=} YbX$$

To solve this, Y must start with a, and X end with b (or be empty) so we have (replacing X by Xb and Y by aY, re-using the same symbols for the new variables)

$$aXbXbb \stackrel{?}{=} aYbXb \quad \longrightarrow \quad XbXb \stackrel{?}{=} YbX$$

Now (the new) X must end with b again, so we have

$$XbbXbb \stackrel{?}{=} YbXb \quad \longrightarrow \quad XbbXb \stackrel{?}{=} YbX$$

We can repeat this as many times as we like, and get

$$Xb^iXb \stackrel{?}{=} YbX$$

for $i \geq 1$. So we had better look at Y instead: Y must be longer than X for the two sides to end up the same length, in fact we need Y to have the same length as Xb^i, and also Y must have prefix $Xb\ldots$, so in fact Y must be Xb^i. This gives

$$Xb^iXb \stackrel{?}{=} Xb^ibX \quad \longrightarrow \quad Xb = bX$$

and now (using the first example) we have found all solutions (if we remember what substitutions we made along the way):

$$X = b^j, Y = ab^j \text{ with } j \geq 0.$$

That was all pretty ad-hoc. Here is the general problem.

Problem: given any expression $U \stackrel{?}{=} V$, where U, V are finite strings (words) in letters and variables

1. answer yes/no if there is any solution
2. find all solutions
3. describe all solutions in an efficient way.

What we are really asking for is an *algorithm* to answer these 3 problems that works on any input $U \stackrel{?}{=} V$, or a proof that there is no such algorithm (and the problems are *undecidable*).

A seemingly harder problem is to allow letters (and variables) to have *inverses*, so the letters are $a, a^{-1}, b, b^{-1}, \ldots$ and two words are *equal* if they reduce to the same strings after removing all cancelling pairs aa^{-1}, etc. For example,

$$aXXb \stackrel{?}{=} YbX$$

This time X could start with a^{-1} or not, and end with either b or b^{-1}, or maybe Y ends with $bX^{-1}b^{-1}$ (see if you can find all solutions to this — you may as well insist your solutions are themselves reduced words, since you will reduce them after substituting anyway).

In the 1970s, Makanin constructed a (really complicated) algorithm which decided problem 1, first for the case without inverses, and then for the case with inverses. Later, Razborov in his thesis answered problem 2 (and 3), claiming all solutions could be encoded using a diagram (now called a *Makanin-Razborov diagram*).

The complexity of these algorithms was shown to be pretty bad (not primitive recursive).

In this talk I will describe a new approach to answer these problems, using an algorithm which runs in PSPACE (polynomial space), and yields a description of the set of all solutions (in reduced words) to any equation as a relatively simple formal language.

The proof involves some pretty cool ideas developed by several researchers including Plandowski, Jeż and Diekert, based on data compression and some probabilistic arguments. In my colloquium talk I will try to motivate and explain the problem (if this abstract did not already do it) and how it relates to some questions in logic, describe the formal language class we prove the set of solutions belongs to (which comes from ideas in biology), and give a rough idea of the proof.

In my subsequent seminar talk I will give more rigorous details of the proof, including why I insisted on (confusingly) using the same names for the variables after they got substituted, and how we reduce the problem in free groups (equations where letters have inverses and can cancel) to a problem in free monoids (no inverses) subject to certain *rational constraints*.

The result is joint work with Laura Ciobanu (Neuchâtel) and Volker Diekert (Stuttgart) and a preprint of the work is at `http://arxiv.org/abs/1502.03426`

Biography: Murray did an undergraduate applied science degree at LaTrobe–Bendigo 1991-1993, then a postgrad diploma and coursework masters at Melbourne Uni, getting an APA to do a PhD at Melbourne with Walter Neumann, working on geometric and automatic group theory. He then got postdocs at Texas A&M, Tufts (Boston), St Andrews (Scotland) then lecturing positions at Wollongong, Stevens (USA) and Queensland before coming to Newcastle in 2011. In 2011 he scored a Future Fellowship from the ARC to work on algorithmic and computational problems in infinite group theory.

Seminar: Solving equations in free groups and free monoids with rational constraints – more details

Speaker: Dr Murray Elder, The University of Newcastle

Abstract: My second talk will give more details of the proof of our result – the set of all solutions to an equation over a free group is an EDT0L language, and can be found in space $O(n \log n)$.

- reduction from an equation over a free group to an equation over a free monoid with involution with rational constraints
- extended equations and partial commutation
- construction of a finite graph encoding all solutions
- two propositions: any path in the graph from an initial to final vertex is a solution; and any solution is realised by a finite path in the graph from an initial to final vertex
- algorithm to prove the second proposition – block and pair compression.