Bootstrapping the next generation of mathematical social machines

Ursula Martin*, Alison Pease, Joseph Corneli
Oxford University, University of Dundee, University of Edinburgh

Abstract

Tim Berners-Lee defines social machines to be a class of systems "in which the people do the creative work and the machine does the administration." Here, we note that the ansatz in computational creativity – a subfield of artificial intelligence – is that, in the future if not already, computational systems and agents will also contribute directly to creative work. But it can be presumed that we will use standard social machines to develop the next generation 'target' systems, whatever they may be.

The domain of mathematics is a particularly interesting and practical place to explore a range of issues in AI and other aspects of computing. Contemporary mathematical practice intersects several areas of computing practice and theory, ranging from computer-mediated communication to formal computational modelling of mathematical argument.

We ask: How will we use the various existing social machines to bootstrap the next edition? How can the idea of programming languages help?

Bryan Birch, Professor Emeritus at the Oxford Mathematical Institute, is credited with the off-handed remark that he programmed in a very high-level programming language called 'graduate student'.[1]

This anecdote is a schematic match to David Murray-Rust and David Robertson's [3] idea of bootstrapping social machines. These authors partition the overall information ecosystem into 'development' and 'target' machines. For example, a certain set of tools and collaboration strategies went into building the first version of Mediawiki and creating the first few articles in Wikipedia. With this system, one of the world's most popular websites was bootstrapped.

Tim Berners-Lee [2] defines social machines to be a class of systems "in which the people do the creative work and the machine does the administration." The ansatz of *computational creativity* is that, in the future if not already, computational systems and agents will also contribute directly to creative work. (These days, more edits to Wikipedia are made by bots than by people.)

The domain of mathematics is a particularly interesting – and practical – place to explore a range of issues in AI and other branches of computing. Many

*Corresponding author, Ursula.Martin@cs.ox.ac.uk

[1] http://mathoverflow.net/questions/11084/what-programming-languages-do-mathematicians-use

mathematicians engage in computer-mediated communication – via email, blogs, wiki, Q&A sites, etc. They use and create digital libraries, also, to an increasing extent – as traces of these communications, and resulting publications, are stored for later use. Computer algebra and other branches of symbolic computing have given rise to their own programming languages, like those developed by Wolfram Research. Theorem proving, after several decades largely within the ægis of computer science, is currently finding an archipelago of use cases within mainstream mathematics. And since its days of origin, mathematics has been closely tied to argumentation: argumentation theory is increasingly capable of modelling the way people do mainstream mathematics.

The next generation of mathematical social machines will need to be conversant with the core ideas in these several modes of engagement. They will also – fundamentally – need to be conversant with the language of mathematics. We have been developing a new strategy for representing mathematical discussions, in which 'The Cayley graph of group G' becomes an object in its own right; the relationship between the proposition P and the proposition 'P is difficult to prove' is made explicit; and in which Lakatos-style conjectures, refutations, and repair are modelled. A range of mathematical predicates like `stronger`, `not`, `equivalent`, `weaker`, `has_property`, and `case_split` are supported, along with meta-level predicates like `goal`, `strategy`, and `auxiliary` – which are used for guiding proofs. Value judgements such as `easy`, `plausible`, `beautiful`, and `useful`, along with performatives like `agree`, `assert`, `challenge`, `define`, `query`, `retract`, and `suggest` round out this proto-language.

We envision extensions to this lexicon that cover the technical (social mechanical) aspects of mathematical dialogues, surveyed above, with robust interfaces to contemporary social machines (like Wikidata, MathOverflow, etc.). Related work in the programming language literature includes the language Dog [1], which was designed to make it easy to coordinate online services. Service oriented architecture (SOA), in which some responsibility for service assembly is handed over to the computer, is also relevant. 'Scope' and 'context' present particularly interesting challenges. As we imagine was also the case for Birch's graduate student, our language wil have to learn – grow, and develop – by doing.

References

[1] Salman Ahmad and Sepandar Kamvar. The Dog Programming Language. In Shahram Izadi, Aaron Quigley, Ivan Poupyrev, and Takeo Igarashi, editors, *Proc. of the 26th annual ACM symposium on User interface software and technology*, pages 463–472. ACM, 2013.

[2] Tim Berners-Lee and Mark Fischetti. *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor.* Harper Information, 2000.

[3] Dave Murray-Rust and Dave Robertson. Bootstrapping the next generation of social machines. In *Crowdsourcing*, pages 53–71. Springer, 2015.