Interview with Murdoch J. Gabbay and Andrew M. Pitts 2019 Alonzo Church Award Recipients

Luca Aceto
Gran Sasso Science Institute, L'Aquila, Italy
ICE-TCS, Department of Computer Science,
Reykjavik University, Iceland
`luca.aceto@gssi.it` and `luca@ru.is`

The 2019 Alonzo Church Award committee (which consisted of Thomas Eiter, Javier Esparza, Radha Jagadeesan, Catuscia Palamidessi and Natarajan Shankar) have selected Murdoch J. Gabbay and Andrew M. Pitts for the 2019 Alonzo Church Award for Outstanding Contributions to Logic and Computation[1]. That award was established in 2015 by the ACM Special Interest Group for Logic and Computation, the European Association for Theoretical Computer Science, the European Association for Computer Science Logic, and the Kurt Gödel Society and is given for an outstanding contribution represented by a paper or small group of papers within the past 25 years.

The citation for the 2019 Alonzo Church Award states that Andrew and Jamie receive the award 'for introducing the theory of nominal representations, a powerful and elegant mathematical model for computing with data involving atomic names.' In particular, the nomination for the Alonzo Church Award singled out the following two papers:

- Murdoch J. Gabbay and Andrew M. Pitts, 'A new approach to abstract syntax with variable binding', Formal Aspects of Computing 13(3):341–363, 2002[2]; and
- Andrew M. Pitts, 'Nominal logic, a first order theory of names and binding', Information and Computation 186(2):165–193, 2003[3].

[1]`https://siglog.org/awards/alonzo-church-award/`
[2]`https://link.springer.com/article/10.1007/s001650200016`
[3]`https://doi.org/10.1016/S0890-5401(03)00138-X`

For the conference version of the first article, Andrew and Jamie will also be receiving the Test-of-Time Award from LICS 1999, which will be presented to them at LICS 2019 in Vancouver.

The award recipients kindly agreed to answer some questions of mine via email. You can find the transcript of the interview below. My questions are labelled with LA, Andrew's answers with AP and Jamie's with JG. I hope that you'll enjoy reading their insights and the story of their award-receiving work as much as I did myself.

The interview

LA: You are receiving the 2019 Alonzo Church Award for Outstanding Contributions to Logic and Computation as well as the Test-of-Time Award from LICS 1999 for your invention of nominal techniques to provide a semantic understanding of abstract syntax with binding. Could you briefly describe the history of the ideas that led you to use the permutation model of set theory with atoms due to Fraenkel and Mostowski to represent name abstraction and fresh name generation? What were the main inspirations and motivations for your work? In your opinion, how did nominal techniques advance the state of the art at that time?

AP: I have had a long-standing interest in the mathematical semantics of programming language features that restrict resources to a specific scope, or hide information from a program's environment; think local mutable state in languages like OCaml, or channel-name restriction in the pi-calculus. When Ian Stark was doing his PhD with me in the 90s we tried to understand a simple instance: the observable properties of higher-order functions combined with dynamically generated atomic names that can be tested for equality, but don't have any other attribute — we called this the 'nu-calculus'. Ian gave a denotational semantics for the nu-calculus using Moggi's monad for modelling dynamic allocation. That monad is defined on the category of pullback-preserving functors from the category of injective functions between finite ordinals to the category of sets. This functor category was well-known to me from topos theory, where it is called Schanuel's topos and hosts the generic model of a geometric theory of an infinite decidable set. A few years later, when Jamie joined me as a PhD student in 1998, I suggested we look at the Schanuel topos as a setting for initial algebra semantics of syntax involving binding operations, modulo alpha-equivalence. I think Jamie prefers set theory over category theory, so he pushed us to use another known equivalent presentation of the Schanuel topos, in terms of continuous actions of the group of permutations of the set $\mathbb{N}$ of natural numbers (topologized as a subspace of the product of countably many copies of $\mathbb{N}$). In this form there is an obvious connection with the cumulative hierarchy of sets (with atoms) that are hereditarily finitely supported with respect to the action of permuting atoms. This universe of sets was devised by Fraenkel and Mostowski in the first part of the twentieth century to model ZFA set theory without axioms of choice. Whether one emphasises set theory or category theory, the move to making permutations of names, rather than injections between sets of names, the primary concept was very fruitful. For example, it certainly makes higher-order constructions (functions and powersets) in the topos/set-theory easier to describe and use. We ended up with a generic construction for name-abstraction modulo alpha-equivalence compatible with classical higher-order logic or set theory, so long as one abstains from unrestricted use of choice.

JG: At the time it wasn't an idea to consider names as elements of a distinctive datatype of names, with properties just like other datatypes such as the natural numbers Nat. If we want to add 1 to 1, we take 1:Nat and invoke the 'plus' function, which is a specific thing associated to Nat; so why not abstract a in x by assuming a:Atm (where Atm is a distinct thing in our mathematical universe) and x:X and invoking a function 'abstract', which is a thing associated to Atm? We unfolded the implications of this idea in set theory and rediscovered FM sets. I was inspired by the way I saw mathematics built up in ZF set theory as an undergraduate, starting from a simple basis and building up the cumulative hierarchy. When I saw the chance to do this for a universe with names, I jumped at the chance. It turns out FM sets are not required. Nominal techniques can be built in ZFA set theory, which contains more functions and permits unrestricted choice.

LA: Over the last fifteen years, nominal techniques have become a fundamental tool for modelling locality in computation, underlying research presented in over a hundred papers, new programming languages and models of computation. They have applications to the syntax and semantics of programming languages, to logics for machine-assisted reasoning about programming-language semantics and to the automatic verification of specifications in process calculi. Variations on nominal sets are used in automata theory over infinite alphabets, with applications to querying XML and databases, and also feature in work on models of Homotopy Type Theory. When did it dawn on you that you had succeeded in finding a very good model for name abstraction and fresh name generation, and one that would have a lot of impact? Did you imagine that your model would generate such a large amount of follow-up work, leading to a whole body of work on nominal computation theory?

AP: No, to begin with I was very focussed on getting better techniques for computing and reasoning about syntax with bound names. But that only represents a part of the current broad landscape of nominal techniques, the part that mainly depends on the mathematical notion of 'finite support' (a way of expressing, via name-permutation, that an object only involves finitely many names). Independently of us, some people realised that a related notion of finiteness, 'orbit-finiteness' (which expresses that an object is finite modulo symmetries) is crucial for many applications of nominal techniques. I am referring to the work of Montanari and Pistore on pi-calculus and HD automata using named sets (yet another equivalent of the Schanuel topos) and the work on automata theory over infinite alphabets (and much else besides) using 'sets with atoms' by the Warsaw group (Bojanczyk, Klin, Lasota, Torunczyk,...). The latter is particularly significant because it considers groups of symmetry for atoms other than the full permutation group (in which the only property of an atom preserved under symmetry is its identity).

JG: Yes, I did. Nobody could anticipate the specific applications but I knew we were on to something, which is why I stayed on to build the field after the PhD. The amount of structure was just too striking. This showed early: e.g. in the equivariance properties, and the commutation of nominal atoms-abstraction with function-spaces. When I sent the proof of this property to Andrew, at first he didn't believe it! I had a sense that there was something deep going on and I still do.

LA: What is the result of yours on nominal techniques you are most proud of? And what are your favourite results amongst those achieved by others on nominal computation?

AP: Not so much a specific result, but rather a logical concept, the freshness quantifier (which we wrote using an upside down 'N' – N stands for 'New'). In informal practice when reasoning about syntax involving binders, one often chooses some fresh name for the bound variable, but then has to revise that choice in view of later ones; but fortunately any fresh name does as well as some particular one. This distinctive 'some/any' property occurs all over the place when computing and reasoning about languages with binders and the freshness quantifier formalises it, in terms of the freshness ('not in the support of') relation and conventional quantifiers. For the second part of your question I would choose two things. One is the work by Jamie with Fernandez and Mackie on nominal rewriting systems, which won the PPDP Most Influential Paper 10-year Award in 2014[4]. The second is the characterisation of orbit-finite sets with atoms in terms of 'set-builder expressions'—see Klin et al, 'Locally Finite Constraint Satisfaction Problems', Proc. LICS 2015)[5]; it's a nice application of the classical model theory of homogeneous structures with interesting applications for languages that compute with finite structures.

JG: Thanks for asking. Aside from the initial papers, my work on nominal rewrit-

[4]https://www.sciencedirect.com/science/article/pii/S0890540106001635

[5]https://www.mimuw.edu.pl/~szymtor/papers/locfin.pdf

ing with Fernandez has probably had most impact. However, I am rather fond of the thread of research going from Nominal Algebra, through the axiomatisation of substitution and first-order logic and the characterisation of quantification as a limit in nominal sets, and on to Stone duality. It's a mathematical foundation built from a nominal perspective of naming and quantification and I hope that as the state of the art in nominal techniques advances and broadens, it might prove useful. Andrew's book has been helpful in marking out nominal techniques as a field. I also agree with Andrew that orbit-finiteness and the applications of this idea to transition systems and automata, is important. I like the automata work for another concrete reason: nominal techniques were discovered in the context of names and binding in syntax, which has bequeathed a misconception that nominal techniques are only about this. The Warsaw school of nominal techniques gives an independent illustration of the other applications of these ideas.

LA: Twenty years have passed since your LICS 1999 paper and the literature on variations on nominal techniques now contains over a hundred papers. Do you expect any further development related to the theory and application of nominal techniques in the coming years? What advice would you give to a PhD student who is interested in working on topics related to nominal computation today?

AP: For the purpose of answering your question, let's agree to divide LICS topics into Programming Languages and Semantics (PLS) versus Logic and Algorithms (LAS). (So long as we don't think of it as a dichotomy!) Then it seems to me that applications of nominal techniques to LAS are currently in the ascendant and show no sign of slowing down. My own interests are with PLS and there is still work to be done there. In particular, I would like better support for using nominal techniques within the mainstream interactive theorem proving systems: we have the Nominal Package of Urban and Berghofer for classical higher-order logic within Isabelle (which led to Urban and Tasson winning the CADE Skolem Award in 2015), but nothing analogous for systems based on dependent type theory, such as Agda, Coq and Lean. Recent work of Swan (arXiv:1702.01556) gives us a better understanding of how to develop nominal sets within constructive logic; but I have yet to see a dependent type theory that both corresponds to some form of constructive nominal logic under Curry-Howard and is sufficiently simple that it appeals to users of systems like Coq who want to mechanise programming language meta-theory in a nameful style. Really, I would like the utility of the FreshML programming language that Jamie, Mark Shinwell and I proposed in 2003 (and which Mark implemented as a patch of OCaml) restricted to total functional programming in the style of Agda; but I don't quite know how to achieve that.

JG: Yes. We are far from understanding nominal techniques and the field has a lot of life and will continue to surprise. I've always believed that. A key stickingpoint right now is implementations. I wrote a paper about this recently, on equivariance and the foundations of nominal techniques. One point in the paper is a sketch for a next-generation nominal theorem-prover (based on ZFA + equivariance). I'd like to see this carried out, so if anybody reading this is interested then please be in touch. I'd also like to see nominal techniques implemented as a package in a language like Haskell, ML, or even Python! If we can get this stuff into the working programmer's toolbox, in a way that just works and does not require special configuration, then that would be helpful. I suspect that nominal techniques as currently presented in the maths papers, might not fit into a programming language at the moment. The theory is too strong and may need to be weakened first. We need a subset of nominal techniques weak enough to squeeze into an existing language, yet expressive enough for interesting applications. Some general advice, specifically for the PhD student. If you have an idea which most people around you don't understand, consider this may be a gap in the collective imagination. There can be peer pressure when faced by incomprehension to blame yourself, back down, and think about something else. By all means do this, but only if you yourself judge it right to do so.

LA: Is there any general research-related lesson you have learnt in the process of working on nominal techniques?

AP: On the one hand, don't lose sight of what application your theory is supposed to be good for; but on the other hand, let beauty and simplicity be your guide.

JG: Yes:

- Proving stuff is 30% of the work; convincing people is 70%.
- It's the basic ideas that are hard, not the complicated theorems.
- Competence and imagination are orthogonal.
- It doesn't have to be complex to be clever.
- Elegant + applicable is a potent combination.
- Seek out good listeners. Give up quickly on bad ones. Try to be a good listener.
- Other people have a lot to teach you, but it might not be the things you expected.
- Writing papers is fun.

LA: Thanks to both of you for your willingness to answer my questions and congratulations for the awards you will be receiving this summer!