Nominal Algebra

Murdoch J. Gabbay
Heriot-Watt University
Riccarton, Edinburgh EH14 4AS
Scotland, Great Britain
murdoch.gabbay@gmail.com

Aad Mathijssen
Technische Universiteit Eindhoven
P.O. Box 513, 5600 MB Eindhoven
The Netherlands
A.H.J.Mathijssen@tue.nl

Universal algebra [5, 9, 4] is the theory of equalities $t = u$. It is a simple framework within which we can study mathematical structures, for example groups, rings, and fields. It has also been applied to study the mathematical properties of *mathematical truth* and *computability*. For example boolean algebras correspond to classical truth, heyting algebras correspond to intuitionistic truth, cylindric algebras correspond to truth in the presence of predicates as well as propositions, combinators correspond to computability, and so on.

Informal mathematical usage and notation often involve *binding*. In many cases, this involves *freshness* ('does not occur free in') and *α-equivalence* in the presence of *meta-variables*. For example:

- λ-calculus: $\lambda x.(tx) = t$ — if x is fresh for t.
- π-calculus: $(\nu a.P) \mid Q = \nu a.(P \mid Q)$ — if a is fresh for Q.
- First-order logic: $(\forall x.\phi) \wedge \psi = \forall x.(\phi \wedge \psi)$ — if x is fresh for ψ.

Here t, P, Q, ϕ and ψ are meta-variables ranging over concrete terms.

Now take any binder $\xi \in \{\forall, \lambda, \nu\}$. Then:

- $(\xi x.t)[y \mapsto u] = \xi x.(t[y \mapsto u])$ — if x is fresh for u.
- α-equivalence: $\xi x.t = \xi y.(t[x \mapsto y])$ — if y is fresh for t.

Nominal terms [15, 16] are a syntax designed to naturally express binding, freshness and α-equivalence in the presence of these meta-variables. This paper studies **Nominal Algebra** (**NA**), the theory of *equality* between nominal terms.

Nominal algebra permits direct and intuitive axiomatisations of systems with binding, and yet we keep the flavour of 'normal algebra'.

In other work we apply nominal algebra to investigate specific axiomatic systems for truth and computability [6, 7]. In this paper we present the syntax of nominal algebra, its derivation rules, and a sound and complete semantics in nominal sets.

The principal distinguishing feature of nominal algebra is to have two kinds of variables; **atoms** and **unknowns**. In this way it models the instantiation behaviour of meta-variables. In informal notation instantiating t to x in $\lambda x.t$ yields $\lambda x.x$, reflected formally by a syntactic equality $(\lambda[a]X)[a/X] \equiv \lambda[a]a$. NA is a *logical* system in which metavariables are explicitly represented by unknowns, and variables by atoms. We may thus reflect binding in logic in a beautiful and direct fashion in our axioms. For example β-reduction may be

represented as an axiom $(\lambda[a]X)Y = X[a \mapsto Y]$. Here $X[a \mapsto Y]$ is an *explicit substitution* satisfying further axioms which we also consider.

The properties listed above translate very directly as follows:

- $a\#X \vdash \lambda[a](Xa) = X$.
- $a\#Y \vdash (\nu[a]X) \mid Y = \nu[a](X \mid Y)$.
- $a\#Y \vdash (\forall[a]X) \wedge Y = \forall[a](X \wedge Y)$.
- $a\#Y \vdash ([a]X)[b \mapsto Y] = [a](X[b \mapsto Y])$.
- $b\#X \vdash [a]X = [b](X[a \mapsto b])$.

In spite of the simplicity and directness of this approach of explicitly representing metavariables in the syntax as X and Y above, nominal algebra turns out to have interesting and quite subtle derivations and semantics, and specific nominal algebra theories display all the richness of structure of the semantics represented by the syntax they so resemble.

Nominal algebra differs from higher-order algebra [11], because it directly represents metavariables in its syntax. It loses no expressivity since β-reduction may be axiomatised. In certain specific cases, nominal algebra theories may even be better-behaved; for example unification in the core nominal algebra theory is decidable, whereas a corresponding unification problem in higher-order algebra is not [16].

Because of the explicit representation of metavariables, nominal algebra is different from other approaches based on λ-terms, such as the theory of contexts [10] and vanilla simply-typed λ-calculus [12, Figures 6 and 7].

Nominal algebra differs from Nominal Logic [13] in spite of a shared semantics in nominal sets [8]; nominal logic does not directly use nominal terms, as nominal algebra does, and the judgement of freshness in nominal algebra is syntax-directed and thus always decidable whereas that of nominal logic is semantic and may not be, depending on the theory [13, p.175 '*Freshness*'].

Nominal algebra differs from some other algebraic methods for axiomatising specific theories, such as lambda-abstraction algebras [14] for the λ-calculus and cylindric algebras [4, 1] for predicate logic. This is because the richer syntax of nominal terms (compared to lambda-abstraction algebras and cylindric algebras) is more expressive on open terms. Though there is no difference in derivability between the relevant nominal algebra theories and lambda-abstraction algebras or cylindric algebras *on closed terms*, on open terms the nominal algebra theory is strictly more expressive. Algebra is properly the study of equalities on open terms because they represent general truths within the algebraic system rather than specific facts, so an increase in power on open terms is significant.

Finally, nominal algebra is different from approaches based on combinators [3, 2]. First, the combinator-based methods is more foundational/semantic than proof-theoretic/algebraic (the flavour of this work), because the extreme self-reflective power of untyped combinators makes contradictions hard to avoid. Second, as mentioned above the expressive power of nominal terms allows variables to be represented explicitly in a way which combinators deliberately eliminate.

References

[1] H. Andréka, I. Németi, and I. Sain. Algebraic logic. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, 2nd Edition*, volume 2, pages 133–249. Kluwer, 2001.

[2] H. Barendregt, W. Dekkers, and M. Bunder. Completeness of two systems of illative combinatory logic for first-order propositional and predicate calculus. *Archive für Mathematische Logik*, 37:327–341, 1998.

[3] H. P. Barendregt. *The Lambda Calculus: its Syntax and Semantics (revised ed.)*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984.

[4] S. Burris and H. Sankappanavar. *A Course in Universal Algebra*. Springer, 1981. Available online.

[5] P.M. Cohn. *Universal Algebra*. Harper and Row, New York, 1965.

[6] Murdoch J. Gabbay and Aad Mathijssen. Capture-avoiding substitution as a nominal algebra. In *ICTAC'2006*, 2006.

[7] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order logic. In *PPDP '06: Proceedings of the 8th ACM SIGPLAN symposium on Principles and Practice of Declarative Programming*, pages 189–200, New York, NY, USA, 2006. ACM Press.

[8] Murdoch J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13(3–5):341–363, 2001.

[9] J. Loeckx, H.D. Ehrich, and M. Wolf. *Specification of Abstract Data Types*. Wiley, 1996.

[10] Marino Miculan. Developing (meta)theory of lambda-calculus in the theory of contexts. *ENTCS*, 1(58), 2001.

[11] B. Möller, A. Tarlecki, and M. Wirsing. Algebraic specification of reachable higher-order algebras. In *ADT 1987*, volume 332 of *LNCS*, pages 154–169, 1987.

[12] Lawrence C. Paulson. The foundation of a generic theorem prover. *Journal of Automated Reasoning*, 5(3):363–397, 1989.

[13] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186(2):165–193, 2003.

[14] Antonino Salibra. On the algebraic models of lambda calculus. *Theoretical Computer Science*, 249(1):197–240, 2000.

[15] C. Urban, A. M. Pitts, and Murdoch J. Gabbay. Nominal unification. In *CSL'03 & KGC*, volume 2803 of *LNCS*, pages 513–527, 2003.

[16] C. Urban, A. M. Pitts, and Murdoch J. Gabbay. Nominal unification. *Theoretical Computer Science*, 323(1–3):473–497, 2004.