Quantifier rules in reductive proof using nominal semantics

Murdoch J. Gabbay (reporting on work with Claus-Peter Wirth)

http://www.gabbay.org.uk

Abstract: Reductive proof-search tries to reduce a goal to tautologies. In the presence of quantifiers this becomes a complex design problem by which proof-theory shades into 'proof-engineering'. There is no single right answer here, but there are a lot of practical problems with strong roots in theory. In this work we consider a nominal semantics for this design space. The reduction in complexity is striking, and we get an elementary account of reductive proof-search with quantifiers.

1 The problem

It is not enough to study proof in principle; we may also want to prove things in practice. This means creating notions of derivation that are succinct and amenable to automation. We concentrate on *reductive*, *analytic*, or *backward* proof-search, reducing goals to assumptions. Examples include sequent, tableau, matrix, and indexed-formula-tree systems.

The quantifier $\forall$ is what interests us. It generates a pair of rules, called *left-intro* and *right-intro*, or $(\delta\forall)$ and $(\gamma\forall)$ rules respectively. Intuitively:

1. left-intro/$(\gamma\forall)$ means "$\forall x.\phi$ implies $[r/x]\phi$ for any r", where here $[r/x]$ is the usual capture-avoiding substitution of r for x (call r a **witness**).
2. right-intro/$(\delta\forall)$ means "$[r/x]\phi$ for some *sufficiently generic* r implies $\forall x.\phi$".

$(\gamma\forall)$ has obvious potential for branching and we delay the choice of witness as long as possible by introducing variables called *existential variables*, *meta-variables*, or (in tableaux) *free variables*. These are variables whose instantiation transforms the proof as a whole. We use variables X from nominal terms to do this (no surprise to the expert in nominal techniques; X was a unification unknown in [20]).

Concerning $(\delta\forall)$: what should 'sufficiently generic' mean? The standard rule takes a fresh entity variously called a *fresh constant*, a *fresh variable*, an *eigenvariable*, or *parameter*. Here it is in sequent style:

$$\frac{\Gamma \vdash \psi, \Delta \quad (x \text{ fresh for } \Gamma, \Delta)}{\Gamma \vdash \forall x.\psi, \Delta}\,(\forall\mathbf{R})/(\delta^-\forall)$$

This rule is inefficient because x is *unnecessarily generic*; choosing x 'completely fresh' does not record—and cannot take advantage of—information about what variables existed when x was created.

An industry exists devising rules to prove $\forall a.\phi$ more efficiently, and it is worthwhile to list some of it: Fitting's original free-variable δ-rule [7, Section 7.4]; then its 'liberalised' version δ^+ (introduced in [16], and treated in [8, Section 7.4]); δ^{+^+} [2]; δ^* [1];[1] δ^{**} [3];[2] and δ^ε [15, Section 4.1].

[1]This had error corrected in [4, Subsection 5.3].

[2]This also had errors, also corrected in [4, Subsection 5.4].

So in the quest for efficiency, inference systems have developed interesting kinds of names and binding, and there is a direct connection between recognising how names in derivations interact (and when they must be generated, and when they may be thrown away) and devising efficient quantifier rules.[3] This kind of thing is hard to get right, errors have been made, and aside from work by Wirth reported in [21], no semantics has been available to aid understanding.

This abstract reports on cutting-edge research in the application of two nominal tools to reductive proof-search: *permissive-nominal terms* from [6] and *nominal sets* semantics from [14] (surveys in [9, 11]).

Technical details are elided. In this abstract we give a flavour of how this rather substantial body of mathematics hangs together. If pressed to describe this work in a sentence, it is this: we have an elementary explanation of the variables and meta-variables typically found in proof-search, as nominal atoms and unknowns, *and* of the proof-search rules listed above; and if you can get past the unfamiliar nominal-ness of the semantics, the technical difficulty threshold is quite low.

2 Sketch of the syntax

Fix disjoint countably infinite sets of **atoms** a, b, c and **variables/unknowns** X, Y, Z.

Atoms a are variables in goals and resemble the *parameters* or *eigenvariables* found in the literature. This is the entity introduced by the $(\forall\mathbf{R})$ rules of sequent systems. Variables X are proof-search variables; these display complex 'nominal' behaviour but have the effect of Skolem terms, without extending the signature or introducing functions. X corresponds to the *dummy* variable of [19] and [18] and the *free variable* of [8], the *meta-variable* of planning and constraint solving, and the *free γ-variable* of [21].

Assume constants $\bot$, $\top$, $\neg$, $\wedge$, and $\forall$ and a simple type system which we elide.

Then **terms** are just $r ::= a \mid X \mid \mathsf{f} \mid r'r \mid [a]r$.[4]

[3]Speedups can be significant. The (δ^+) of [16] allows exponential speedup relative to (δ^-) [8], and (δ^{+^+}) [2] allows further exponential speedup relative to (δ^+) [1, Section 3].

[4]A white lie: X is moderated as $\pi{\cdot}X$. See [20, 6].

This looks familiar (variables; meta-variables; constants; application; abstraction) but substitution for X is capturing and the semantics of $[a]r$ is nominal atoms-abstraction instead of functional abstraction. So, the underlying semantics is different, non-functional, and 'first-order'; for details see [13] which applies this to Henkin-style semantics for higher-order logic.

Atoms are not constant symbols; models are subject to a permutation symmetry group of atoms, so atoms are special symmetric elements, translated specially in the denotation. Constant symbols are interpreted arbitrarily; no special properties are assumed. So $[a]r$ makes sense because the interpretation of a is specific such that atoms-abstraction has meaning; $[\mathsf{f}]r$ makes no model-theoretic sense because there is virtually no restriction on how f is interpreted.

3 Sketch of the proof-rules

Here are the two crucial rules, in sequent style:

$$\frac{\mathcal{C}, X{\uparrow}[a]\neg\phi; \mathcal{H} \vdash \phi[a{\mapsto}X]}{\mathcal{C}; \mathcal{H} \vdash \forall a.\phi}\ (\delta^{\mathsf{x}}\forall\mathbf{R}) \qquad \frac{\mathcal{C}; \phi[a{\mapsto}r] \vdash \psi}{\mathcal{C}; \mathcal{H}, \forall a.\phi \vdash \psi}\ (\delta^{\mathsf{x}}\forall\mathbf{L})$$

$\mathcal{H}$ is a set of predicates. $\mathcal{C}$ is a **maximisation** condition; a set of syntax of the form $X{\uparrow}[a]\phi$. This can be read as an instruction to 'maximise' the truth-value of $\phi[a{\mapsto}X]$. Intuitively, if the value of X makes $\neg\phi[a{\mapsto}X]$ true, then $\forall a.\phi$ must be true. If this reminds the reader of expressing $\forall a.\phi$ as $\phi[a{\mapsto}\epsilon a.\neg\phi]$ [5, page 15] then that is no accident—but here there is no choice made, only a maximisation *condition*.

The nominal semantics makes itself particularly useful because ϕ is a possibly open predicate—it may have free atoms. Nominal semantics allow us to map this open predicate to an open element of a nominal algebra of possibly open truth-values.

The underlying message is that nominal semantics here replace Skolemisation in both syntax and semantics. The price we pay is a notion of 'truth-values algebra with atoms', but this seems not only worthwhile but is in itself interesting.

4 Sketch of the semantics

We just give the flavour of how it works; the particularly dedicated reader can find full details of similar technology applied in abstract algebra in [10]. We assume a Boolean algebra, but elements are nominal and contain free atoms. These atoms can be substituted for; *this substitution is not syntactic*, but an abstract nominal algebraic axiomatisation of substitution following [12]. Next is quantification, which is just an operation on algebra elements related to the quantification operation of *cylindric algebra* [17] but in a nominal context. Atoms are interpreted as themselves and variables X are interpreted with valuations.

Once all this is in place, proving compositionality of the semantics *and* the proof-rules is very easy, because the semantics includes abstract nominal structures which mirror what is done in the syntax. Every 'normal' model (with Skolemisation and choice) can be converted to a nominal model, so this simplicity does not come by absurd restriction of models.

5 Summary

What we have discussed can be accomplished with choice and Skolemisation. But these are powerful, and using such tools has a price; we must manipulate a system with more structure than necessary, and must use many emulations.

Nominal techniques give the benefits of Skolemisation without introducing functions or higher types. Maximisation conditions give the benefit of Hilbert's choice without making choices. The nominal semantics is not hard—we can even import it off-the-shelf from [13, 10]—and allows us to talk about open elements easily and directly, and it all fits together nicely.

References

[1] M. Baaz and C. G. Fermüller. Non-elementary speedups between different versions of tableaux. In *Proceedings of the 4th International Workshop on Theorem Proving with Analytic Tableaux and Related Methods (TABLEAUX'95)*, pages 217–230. Springer, 1995.

[2] B. Beckert, R. Hähnle, and P. H. Schmitt. The even more liberalized delta-rule in free variable semantic tableaux. In G. Gottlob, A. Leitsch, and D. Mundici, editors, *Proceedings of the third Kurt Gödel Colloquium (KGC'93)*, volume 713 of *LNCS*, pages 108–119. Springer, 1993.

[3] D. Cantone and M. Asmundo. A further and effective liberalization of the δ-rule in free variable semantic tableaux. In R. Caferra and G. Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *Lecture Notes in Computer Science*, pages 408–414. Springer, 2000.

[4] D. Cantone and M. Nicolosi-Asmundo. A sound framework for delta-rule variants in free variable semantic tableaux. *Journal of Automated Reasoning*, 38:31–56, 2007.

[5] P. B. David Hilbert. *Grundlagen der Mathematik (volume II)*. Number 50 in Grundlehren der mathematischen Wissenschaften. Springer, second edition, 1970.

[6] G. Dowek, M. J. Gabbay, and D. P. Mulligan. Permissive Nominal Terms and their Unification: an infinite, co-infinite approach to nominal techniques (journal version). *Logic Journal of the IGPL*, 18(6):769–822, 2010.

[7] M. Fitting. *First-order Logic and Automated Theorem Proving*. Texts and monographs in computer science. Springer, 1 edition, 1990.

[8] M. Fitting. *First-order Logic and Automated Theorem Proving*. Texts and monographs in computer science. Springer, 2 edition, 1996.

[9] M. J. Gabbay. Foundations of nominal techniques: logic and semantics of variables in abstract syntax. *Bulletin of Symbolic Logic*, 17(2):161–229, 2011.

[10] M. J. Gabbay. Stone duality for First-Order Logic: a nominal approach. In *Howard Barringer Festschrift*. December 2011.

[11] M. J. Gabbay. Nominal terms and nominal logics: from foundations to meta-mathematics. In *Handbook of Philosophical Logic*, volume 17. Kluwer, 2012.

[12] M. J. Gabbay and A. Mathijssen. Capture-Avoiding Substitution as a Nominal Algebra. *Formal Aspects of Computing*, 20(4-5):451–479, June 2008.

[13] M. J. Gabbay and D. Mulligan. Nominal Henkin Semantics: simply-typed lambda-calculus models in nominal sets. In *Proceedings of the 6th International Workshop on Logical Frameworks and Meta-Languages (LFMTP 2011)*, volume 71 of *EPTCS*, pages 58–75, September 2011.

[14] M. J. Gabbay and A. M. Pitts. A New Approach to Abstract Syntax with Variable Binding. *Formal Aspects of Computing*, 13(3–5):341–363, July 2001.

[15] M. Giese and W. Ahrendt. Hilbert's ϵ-terms in automated theorem proving. In N. Murray, editor, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 1617 of *Lecture Notes in Computer Science*, pages 662–662. Springer, 1999.

[16] R. Hähnle and P. H. Schmitt. The liberalized δ-rule in free variable semantic tableaux. *Journal of Automated Reasoning*, 13(2):211–222, 1994.

[17] L. Henkin, J. D. Monk, and A. Tarski. *Cylindric Algebras*. North Holland, 1971 and 1985. Parts I and II.

[18] S. Kanger. A simplified proof method for elementary logic. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning volume 1 - classical papers on Computational Logic 1957-1966*, pages 364–371. Springer, 1963.

[19] D. Prawitz. An improved proof procedure. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning volume 1 - classical papers on Computational Logic 1957-1966*, pages 159–199. Springer, 1983.

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

[21] C.-P. Wirth. Descente infinie + Deduction. *Logic Journal of the IGPL*, 12(1):1–96, 2004.