Aspects of Nominal Rewriting

Murdoch J. Gabbay

CWI Amsterdam, TU Eindhoven, Netherlands, 2/2005

This talk has 16 slides including the two so far. Please interrupt with questions.

— Jamie 'no time pressure' Gabbay

The issue

Rewriting is a framework/methodology which can formally express computation, logic, and processes. To ‘do rewriting’ it suffices to decide on a grammar for terms, and to write down rules describing how to transform one term into another.

For example:

- The λ-calculus is a rewrite system with terms *blah for λ-terms* and rewrites *blah for reductions*.
- The π-calculus is a rewrite system with terms *blah for π-terms* and rewrites *more blah for reactions*.
- First-order logic is a rewrite system with terms *blah for judgements* and rewrites *more blah for derivation rules, read bottom-up*.

For example: a λ-calculus

Terms are given by: $t ::= a \mid \lambda a.t \mid tt \mid t[a{\mapsto}t]$.

Reductions are given by:

$$(\lambda a.X)Y \leadsto X[a{\mapsto}Y] \qquad (XY)[a{\mapsto}U] \leadsto (X[a{\mapsto}U])(Y[a{\mapsto}U])$$

$$(\lambda b.X)[a{\mapsto}U] \leadsto \lambda b.(X[a{\mapsto}U]) \quad (a \notin FV(X))$$

- X, Y, and U are meta-variables standing for unknown terms (alternative: one rewrite rule for every term, analagous to *axiom schemes* in logic).
- a and b are (what I shall call) names or atoms; variable symbols of the object-language
  (alternative: use meta-variables to represent names).
- Names get abstracted whence capture-avoidance side-conditions (with alternative: use λ-abstraction to represent object-level abstraction).

CRS and HOAS

I may have Jan Willem Klop in the audience, or indeed van Oostrom, van Raamsdonk, or others from the community of Dutch researchers who have developed Combinatory Reduction Systems (CRS) and more generally furthered Higher-order abstract syntax (HOAS).

HOAS assumes some form of λ-term and does rewriting on that. This enables us to use meta-variables to model object-variables, and meta-level binding to model object-level binding. We do have to be a bit careful not to let undecidability of computation in the λ-calculus infect our framework, and we may have to work a bit to pass meta-level unknowns through λ-terms to their intended position.

CRS and HOAS

My life would be easier if I could say now that Nominal techniques are a special case, or a generalisation, or an inverse mapping, or *something* to do with HOAS, and Nominal Rewriting have something to do with CRS.

Indeed it is possible to map CRS into Nominal Rewriting, but the map is via an encoding of the λ-calculus. I feel that is a convergence of functionality, not underlying mechanism.

The broader connections between Nominal Techniques and HOAS are unclear, for the moment. My best explanation so far is a-logic (paper on the web).

Basic idea

- Unknowns (meta-variables) X, Y, Z, U are distinct from atoms (object-level variables) a, b, c.
- There is an abstraction operator $[a]s$ which does not bind, in the sense that substitution of t for X in $[a]s$ is first-order textual substitution and does not avoid capture.
- There is a context of freshness assumptions $a\#X$ in which rewriting takes place. We are *not allowed* to substitute t for X if $a\#t$ is not provable.

I will define 'not allowed' later.

$(\lambda[a]X)$ is 'λ of abstract a in X'. λ is an operator; all abstractors are (for sorts, see later). So similarly, $\nu[a]P$ is 'ν of abstract a in P'.

$$\frac{a\#s_1 \ \cdots \ a\#s_n}{a\#\langle s_1,\ldots,s_n\rangle} \qquad \frac{a\#s}{a\#fs} \qquad \frac{a\#s}{a\#[b]s}$$

$$\frac{}{a\#b} \qquad \frac{}{a\#[a]s} \qquad \frac{\pi^{-1}(a)\#X}{a\#\pi\cdot X}$$

Write Δ for a set of apartness assumptions $a\#X$. Write $\Delta \ \vdash \ a\#s$ when assumptions Δ prove $a\#s$.

$$a\#X \ \vdash \ a\#\langle X,[a]Y\rangle$$

$$a\#X,\ b\#Y \ \vdash \ a\#\langle (a\ b)\cdot X,(b\ c)\cdot Y\rangle$$

π is a atoms-permutation, e.g. $(a\ b)$ swaps a and b. We may use them to rename atoms to avoid capture, e.g. when we deduce equality:

$\approx_\alpha$ Equality-as-a-logical-notion

$$\frac{s_1 \approx_\alpha t_1 \;\cdots\; s_n \approx_\alpha t_n}{\langle s_1,\ldots,s_n\rangle \approx_\alpha \langle t_1,\ldots,t_n\rangle} \quad \frac{s \approx_\alpha t}{fs \approx_\alpha ft} \quad \frac{}{a \approx_\alpha a} \quad \frac{t \approx_\alpha t'}{t' \approx_\alpha t}$$

$$\frac{s \approx_\alpha t}{[a]s \approx_\alpha [a]t} \quad \frac{a\#t \qquad s \approx_\alpha (a\,b)\cdot t}{[a]s \approx_\alpha [b]t} \quad \frac{ds(\pi,\pi')\#X}{\pi\cdot X \approx_\alpha \pi'\cdot X}$$

$ds(\pi,\pi') = \{a \mid \pi(a) \neq \pi'(a)\}$ the difference set.

Write $\Delta \vdash s \approx_\alpha t$ when Δ proves $s \approx_\alpha t$.

$$\begin{array}{rlcl} a,b\#X \vdash & (a\,b)\cdot X & \approx_\alpha & X \\ b\#X \vdash & \lambda[a]X & \approx_\alpha & \lambda[b](b\,a)\cdot X \end{array}$$

Nominal matching/unification algorithms invert these rules and include a substitution step to solve $X \approx_\alpha t$. We omit details...

- Urban, Pitts, Gabbay ‘Nominal Unification’.
- Fernández, Gabbay ‘Nominal Rewriting’, also ‘Extensions of Nominal Rewriting’.

All on the web.

Brief summary

Nominal Techniques typically:

- Separate meta-level unknowns from object-level variable symbols.
- Separate syntactic identity $\equiv$ from α-equivalence , and therefore also binding (α-renaming preserves identity) from abstraction (only preserves α-equivalence).

  α-equivalence is the useful notion of equivalence, we just do not call α-equivalent terms identical.
- Enrich the context with assumptions about freshnesses $a\#X$.
- Enrich terms themselves with permutations suspended on unknowns $\pi \cdot X$ and abstractions $[a]X$.

Write $V(s)$ for the X in s and $A(s)$ for the a in s. Write $V(\nabla)$ for ∇ a set of freshness assertions.

A nominal rewrite rule (over a signature Σ) is a tuple (∇, l, r), we write it $\nabla \ \vdash \ l \rightarrow r$, such that $V(r) \cup V(\nabla) \subseteq V(l)$. We may write $l \rightarrow r$ for $\emptyset \ \vdash \ l \rightarrow r$.

- $a\#X \ \vdash \ (\lambda[a]X)Y \rightarrow X$ is a form of trivial β-reduction.
- $a\#X \ \vdash \ X \rightarrow \lambda[a](Xa)$ is η-expansion.
- $XY \rightarrow XX$ is strange but quite valid.
- $a \rightarrow b$ is a rewrite rule.
- $a\#Z \ \vdash \ X\lambda[a]Y \rightarrow X$ is not a rewrite rule; $Z \notin V(X\lambda[a]Y)$. $X \rightarrow Y$ is also not a rewrite rule.

I'm telling you we can also do explicit substitutions, the π-calculus, and lots of similar cases.

A Nominal Signature Σ is some sorts of atoms , base data sorts s (e.g. $\mathbb{N}$, $\mathbb{B}$), and function symbols f of arity $\tau_1 \rightarrow \tau_2$. If τ_1 is an empty product say f is 0-ary (i.e. a constant) and omit the arrow.

Term sorts are inductively defined by:

$$\tau ::= \nu \mid s \mid \tau \times \ldots \times \tau \mid [\nu]\tau.$$

$\tau_1 \times \ldots \times \tau_n$ is a product sort. $[\nu]\tau$ is an abstraction sort. Terms are defined in the next slide, but first an example:

A nominal signature for a fragment of ML has one sort of atoms $\mathbb{A}$, one sort of data exp, and function symbols with arities

$$\mathtt{var} : \mathbb{A} \rightarrow exp \qquad \mathtt{app} : exp \times exp \rightarrow exp$$

$$\mathtt{lam} : [\mathbb{A}]exp \rightarrow exp \qquad \mathtt{let} : exp \times [\mathbb{A}]exp \rightarrow exp$$

$$\mathtt{letrec} : [\mathbb{A}](([\mathbb{A}]exp) \times exp) \rightarrow exp$$

Extended nominal terms extend freshness contexts with conditions such as $\bullet X$ for 'X is closed' (meaning: $a\#X$ is provable for all a).

With such a condition in the context, we can only substitute t for X if $a\#t$ is provable for all a. (Actually, you can always do the substitution, but if $\bullet a$ gets in the context you can prove anything.) Because t is finite, it suffices to consider all a in t, and one fresh t.

The logic thickens but so long as it stays decidable this is not a problem.

We also extend terms with $\text{И}a.t$. И is not an abstractor, so $\text{И}a.a \not\approx_\alpha \text{И}b.b$. It is not a binder either, so $\text{И}a.a \not= \text{И}b.b$. However, we assume a rewrite rule

$$\text{(F)} \qquad b\#X \vdash \text{И}a.X \rightsquigarrow \text{И}b.(b\ a) \cdot X.$$

И models name-generation, as distinct from name-binding or name-abstraction!

Meta-properties

- Operates on true first-order terms (abstract syntax trees).
- Critical pairs lemma.
- Orthogonal systems are confluent.
- Matching/Unification/Rewriting is (at worst) quadratic; whether it is linear is so far unknown.
- Can express rewriting for syntax-with-binders.

Conclusions

The interplay of the different notions of *binding*, *abstraction*, *name-generation*, closure, and so on, is *non-trivial* and (I think) *illuminating and very interesting*.

We have applied these ideas to logic (Nominal Logic, Fresh Logic, a-logic) and λ-calculi (NEW calculus of contexts). I am in Eindhoven visting MohammadReza Mousavi and Michel Reniers to see if we can apply these ideas to SOS (structured operational semantics).