Nominal Rewriting

Murdoch J. Gabbay
Work with Maribel Fernández and Ian Mackie

February 27, 2004

A rewrite system consists, informally speaking, of a set of terms t with holes (meta-variables) X and rewrite rules $t \rightarrow t'$.

So let t be terms of the λ-calculus with holes:

$$t ::= a \mid tt \mid \lambda a.t \mid X.$$

Forget about what this actually means (of course I know an 'FM-style' approach). Can we give a rewrite rule for η-equivalence?

$$X \rightarrow \lambda a.(Xa)$$

No: this is only valid when x is not in the free variables of X, whatever that means. We need a side-condition. Oops. Was that all my standard theory of rewriting that just flew out the window? I do believe it was.

We have the same problems in other contexts:

$a\#P \vdash P \mid \nu[a]Q \to \nu[a](P \mid Q)$	Scope extrusion
$a\#t \vdash (\lambda a.t)[b\mapsto t'] \to \lambda a.(t[b\mapsto t'])$	Explicit substitution
$b\#t \vdash \lambda a.t =_\alpha \lambda b.(t[a\mapsto b])$	α-equivalence
$b\#t \vdash \lambda a.t =_\alpha \lambda b.((a\ b)\cdot t)$	α-equivalence (sans peine)

Here P, Q, t, and t' are actually X, Y, Z, and Z'. Note apartness conditions $a\#P$ and swappings $(a\ b)\cdot t$.

Super-Maribel, Super-Jamie, and Super-Ian to the rescue!
Fernández, Gabbay, Mackie, “Nominal Rewriting”, submitted. Introduces notion of ‘nominal rewriting’ and proves critical pair theorem.

Nominal Signatures

Fix $\mathcal{S}$ base data sorts typically called s, for example integer, boolean.

A Nominal Signature Σ is:

1. A set of sorts of atoms typically written ν.
2. A set of data sorts typically written δ. δ can be a base data sort or a product:
$$\delta ::= s \mid \delta \times \delta.$$
3. Compound (data) sorts typically written τ are then *defined* by the following grammar:
$$\tau ::= \nu \mid \delta \mid 1 \mid \tau \times \tau \mid [\nu]\tau.$$

Examples of nominal signatures

λ-calculus (1): Add a data sort Λ and constants $\lambda : [\nu]\Lambda \longrightarrow \Lambda$ and $app : \Lambda \times \Lambda \longrightarrow \Lambda$.

λ-calculus (2): Add a data sort Λ and constants $\lambda : \nu \times \Lambda \longrightarrow \Lambda$ and $app : \Lambda \times \Lambda \longrightarrow \Lambda$.

There is surprisingly little functional difference between these two signatures, we may discuss that later. Similarly:

π-calculus (1): Data sort Π and constants $res : [\nu]\Pi \longrightarrow \Pi$, $\mid : \Pi \times \Pi \longrightarrow \Pi$, $in : \nu \times [\nu]\Pi \longrightarrow \Pi$, $\ldots$.

π-calculus (2): Data sort Π and constants $res : \nu \times \Pi \longrightarrow \Pi$, $\mid : \Pi \times \Pi \longrightarrow \Pi$, $in : \nu \times \nu \times \Pi \longrightarrow \Pi$, $\ldots$.

Terms

Fix Σ. For each τ fix $\mathcal{X}_\tau$ term variables X_τ, Y_τ, Z_τ. They will represent meta-level unknowns. For each ν fix $\mathcal{A}_\nu$ atoms $a_\nu, b_\nu, c_\nu, f_\nu, g_\nu, h_\nu, \ldots$. We shall drop the subscripts.

A swapping is a pair $(a_\nu\ b_\nu)$. Permutations π are lists of swappings, write **Id** for the empty list.

Nominal Terms are generated by the following grammar:

$$t ::= a_\nu \mid \pi \cdot X_\tau \mid *_1 \mid \langle t_\tau, t'_{\tau'} \rangle_{\tau \times \tau'} \mid ([a_\nu]t_\tau)_{[\nu]\tau} \mid (f_{\tau \to \delta} t_\tau)_\delta$$

Write that again without subscripts:

$$t ::= a \mid \pi \cdot X \mid * \mid \langle t, t' \rangle \mid [a]t \mid f\, t$$

Write $T(\Sigma, \mathcal{A}, \mathcal{X})$ for the set of terms over a signature Σ.

Write X for **Id** $\cdot X$. Informally $(a\ b) \cdot X$ is "swap a and b in X when instantiated". Informally, $[a]t$ means "t with a bound". This is reflected in

An apartness condition is a pair $a\#X$. Apartness contexts $\Delta, \nabla, \Gamma, \ldots$ are finite sets of apartness conditions. $a\#X$ means "a does not occur in X, when it is instantiated".

Write $V(s)$ and $V(\nabla)$ for 'variables of'.

A nominal rewrite rule is $\nabla \vdash l \rightarrow r$, such that $V(r) \cup V(\nabla) \subseteq V(l)$.
If $\nabla = \emptyset$ we may write $l \rightarrow r$.

Our previous examples are nominal rewrite rules for appropriate signatures. Also:

1. $a\#X \vdash (\lambda a.X)Y \rightarrow X$ is a form of trivial β-reduction.
2. $a\#X \vdash X \rightarrow \lambda a.(Xa)$ is η-expansion.
3. Of course a rewrite rule may define any arbitrary transformation of terms, and may have an empty context, for example $\emptyset \vdash XY \rightarrow XX$.
4. $a\#Z \vdash X\lambda a.Y \rightarrow X$ is not a rewrite rule, because $Z \notin V(X\lambda a.Y)$. $\emptyset \vdash X \rightarrow Y$ is also not a rewrite rule.
5. $\emptyset \vdash a \rightarrow b$ *is* a rewrite rule.

X and Y can be instantiated (more later). a and b cannot. Thus $a \rightarrow b$ does not rewrite b to a. Generally one variable symbol should be like any other. So we consider rules up to permutative renaming:

A set of rewrite rules $\mathcal{S}$ is equivariant when if $R \in \mathcal{S}$ then $R' \in \mathcal{S}$ for all permutative renamings of variable symbols *and* atoms.

$$a\#X \ \vdash \ (\lambda a.X)Y \rightarrow X \quad \text{and} \quad a\#Y \ \vdash \ (\lambda a.Y)X \rightarrow Y,$$
$$a\#X \ \vdash \ X \rightarrow \lambda a.(X a) \quad \text{and} \quad b\#X \ \vdash \ X \rightarrow \lambda b.(X b)$$

A nominal rewrite system $(\Sigma, \mathcal{R})$ consists of: a nominal signature Σ, and an equivariant set $\mathcal{R}$ of nominal rewrite rules over Σ.

Freshness $\#$

$$\frac{a\#s \quad a\#s'}{a\#\langle s, s'\rangle} \qquad \frac{a\#s}{a\#fs} \qquad \frac{}{a\#[a]s}$$

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

$a\#t$ is basically $a \notin fn(t)$. $[a]X$ is the binder, $\pi \cdot X$ permutes.
Because permutations are invertible, we can pull them to the left.

Permutation action

Permutations act on atoms as follows:

$$(a\,b)(a) \stackrel{\text{def}}{=} b \qquad (a\,b)(b) = a \quad \text{and} \quad (a\,b)(c) = c\ (c \neq a, b)$$

$$ds(\pi, \pi') \stackrel{\text{def}}{=} \{n \mid \pi(n) \neq \pi'(n)\}.$$

For example $ds((a\,b), \textbf{Id}) = \{a, b\}$.

We shall write $(a\,b) \cdot t$ for $t \not\equiv X$. This is sugar:

$$(a\,b) \cdot n = (a\,b)(n) \quad (a\,b) \cdot ft = f(a\,b) \cdot t \quad (a\,b) \cdot * = *$$
$$(a\,b) \cdot \langle s, t\rangle = \langle (a\,b) \cdot s, (a\,b) \cdot t\rangle \quad (a\,b) \cdot [n]t = [(a\,b)(n)](a\,b)t$$

α-equivalence

$$\frac{}{* =_\alpha *} \qquad \frac{}{a =_\alpha a} \qquad \frac{ds(\pi,\pi')\#X}{\pi \cdot X =_\alpha \pi' \cdot X} \qquad \frac{s =_\alpha t}{[a]s =_\alpha [a]t}$$

$$\frac{a\#t \quad (a\ b) \cdot s =_\alpha t}{[a]s =_\alpha [b]t} \qquad \frac{s =_\alpha t}{fs =_\alpha ft} \qquad \frac{s =_\alpha t \ \ s' =_\alpha t'}{\langle s, s' \rangle =_\alpha \langle t, t' \rangle}$$

Matching and unification proceed as usual (see below)—except up to $=_\alpha$, in a context of apartness assumptions. Thus $\lambda[a]\lambda[b]\lambda[a]aba =_\alpha \lambda[b]\lambda[a]\lambda[a]bab$.

In the presence of the rule $X \rightarrow X$, $[a]a$ rewrites to $[b]b$.

A problem is a set of apartness conditions $a\#X$ and equality problems $t = t'$. They are solved according to the following algorithm:

$$* \; {}_?{=} \; *, P \to P \qquad \langle l, l'\rangle \; {}_?{=} \; \langle s, s'\rangle, P \to l \; {}_?{=} \; s, l' \; {}_?{=} \; s', P$$

$$fl \; {}_?{=} \; fs, P \to l \; {}_?{=} \; s, P \qquad [a]l \; {}_?{=} \; [a]s, P \to l \; {}_?{=} \; s, P$$

$$[b]l \; {}_?{=} \; [a]s, P \to (a\ b) \cdot l \; {}_?{=} \; s, b\#s, P \qquad a \; {}_?{=} \; a, P \to P$$

$$\pi \cdot X \; {}_?{=} \; \pi' \cdot X, P \to ds(\pi, \pi')\#X, P$$

$$a\#s, P \to \langle a\#s\rangle_{\#sol}, P \qquad (s \not\equiv X)$$

(Matching) $\quad Y \; {}_?{=} \; s, P \overset{Y \mapsto s}{\to} P[Y \mapsto s]$

(Unification) $\quad l \; {}_?{=}_? \; X, P \overset{X \mapsto l}{\to} P[X \mapsto l]$

$\langle a\#s\rangle_{\#sol}$ is the least apartness context entailing $a\#s$.

$$(\lambda[a]X)X' \to X[a \mapsto X'] \qquad \texttt{let}\ a = X'\ \texttt{in}\ X \to X[a \mapsto X']$$

$$\texttt{letrec}\ fa = X'\ \texttt{in}\ X \to X[f \mapsto (\lambda[a]\texttt{letrec}\ fa = X'\ \texttt{in}\ X')]$$

$$(XX')[a \mapsto Y] \to X[a \mapsto Y]X'[a \mapsto Y] \qquad a[a \mapsto X] \to X \qquad a[b \mapsto X] \to a$$

$$a\#Y \ \vdash\ (\lambda[a]X)[b \mapsto Y] \to \lambda[a](X[b \mapsto Y])$$

$$a\#Y \ \vdash\ (\texttt{let}\ a = X'\ \texttt{in}\ X)[b \mapsto Y] \to \texttt{let}\ a = X'[b \mapsto Y]\ \texttt{in}\ X[b \mapsto Y]$$

$$f\#Y, a\#Y \ \vdash\ (\texttt{letrec}\ fa = X'\ \texttt{in}\ X)[b \mapsto Y] \to$$
$$\texttt{letrec}\ fa = X'[b \mapsto Y]\ \texttt{in}\ X[b \mapsto Y]$$

What is a Critical Pair in Nominal Rewriting?

Write $\Delta \vdash s \rightarrow t_1, t_2$ for the appropriate pair of rewrite judgements. Call a valid pair $\Delta \vdash s \rightarrow t_1, t_2$ a peak. If there exists u such that $\Delta \vdash t_1 \rightarrow^* u$ and $\Delta \vdash t_2 \rightarrow^* u$ then say the peak can be joined.

Suppose

1. $R_i = \nabla_i \vdash l_i \rightarrow r_i$ for $i = 1, 2$ are two rules in $\mathcal{R}$ such that $V(R_1) \cap V(R_2) = \emptyset$.
2. p is a position in l_1.
3. $l_1|_p \;{}_?{=}_?\; l_2$ has a solution (Γ, θ), so that $\Gamma \vdash l_1|_p\theta =_\alpha l_2\theta$.

Then call the 'pair of terms'-in-context $\nabla_1\theta, \nabla_2\theta, \Gamma \vdash (r_1\theta, l_1[r_2\theta]_p)$ a critical pair.

Conclusions

Nominal Rewriting is a system very close to first-order rewriting, but the apartness contexts let us avoid variable capture, abstractions let us bind, and swappings let us rename atoms.

This may be useful for expressing rewrite systems on syntax with binding!