PNL: Permissive-nominal logic

Murdoch J. Gabbay
Joint work with Gilles Dowek

June 4, 2010

The role of logic and semantics

Logic is a formal language for modelling mathematical reasoning. Semantics is a formal notion of meaning for logic.

Surprisingly, there is no generally-accepted logic or semantics to model reasoning at the informal meta-level.

We see what we are accustomed to seeing. We are not used to seeing the informal meta-level, as a formal(isable) system. So let me illustrate what I mean by informal meta-level.

The informal meta-level: some examples

Consider η-expansion,

$$(\eta) \qquad \lambda x.(ta) = t \qquad (a \notin fv(t))$$

$\forall$-introduction,

$$\frac{\Phi \vdash \phi \quad (x \notin fv(\Phi))}{\Phi \vdash \forall x.\phi}\,(\forall\mathbf{R})$$

or the π-calculus

$$P \mid \nu a.Q \;=\; \nu a.(P \mid Q) \qquad (a \notin fc(P)).$$

We see a pattern: level 1 (object-level) and level 2 (meta-level) variables, binding, and freshness side-conditions.

Permissive-nominal logic

Permissive-nominal logic (PNL) is designed to cleanly express specifications like those of the last slide.

It has two levels of variable, represented in the syntax formally as atoms a, b, c and unknowns X, Y, Z.

Each unknown X has a permission set $p(X)$ to implement freshness conditions; this is a set of 'permitted atoms' in X. If $a \notin p(X)$ then a is 'fresh for' X.

Here are the same specifications, written formally in PNL:

Some examples, again

η-expansion; here $a \notin p(X)$:

$$\forall X.\ \lambda([a]\mathsf{app}(\mathsf{X}, \mathsf{a})) = \mathsf{X}$$
$$\lambda x.(ta) = t \qquad (a \notin fv(t))$$

$\forall$-introduction; here $a \notin p(P)$:

$$\forall P, Q.\ \mathsf{entails}(\mathsf{P}, \mathsf{Q}) \Rightarrow \mathsf{entails}(\mathsf{P}, \forall([\mathsf{a}]\mathsf{Q})) \qquad \frac{\Phi \vdash \phi \quad (x \notin fv(\Phi))}{\Phi \vdash \forall x.\phi}\,(\forall\mathbf{R})$$

π-calculus; here $a \notin p(P)$:

$$\forall P, Q.\ \mathsf{par}(\mathsf{P}, \nu([\mathsf{a}]\mathsf{Q})) = \nu([\mathsf{a}]\mathsf{par}(\mathsf{P}, \mathsf{Q}))$$
$$P \mid \nu a.Q \ = \ \nu a.(P \mid Q) \qquad (a \notin fc(P)).$$

The permission set $p(P)$ is a set of atoms. $a \notin p(P)$ means that the unknown P may not be instantiated to a term in which the atom a is free. If $a \notin p(X)$ then from

$\forall X.\ \lambda([a]\mathsf{app}(\mathsf{X}, \mathsf{a})) = \mathsf{X}$ you cannot derive $\lambda([\mathsf{a}]\mathsf{app}(\mathsf{a}, \mathsf{a})) = \mathsf{a}$.

How it works

Making this work in a logic and semantics is far from obvious.

There were serious technical challenges to overcome. Many challenges and open questions remain.

The result so far, PNL, can specify systems like first-order logic, the lambda-calculus, the pi-calculus, and arithmetic.

Axiomatisations tend to closely resemble informal specifications. They tend to be finite.

PNL comes with a semantics: permissive-nominal sets. We have proved the following properties:

Properties of permissive-nominal logic

Theories in PNL are sound and complete for models in permissive-nominal sets.

Cut can be eliminated in PNL sequent derivations.

PNL semantics is more 'first-order' than 'higher-order' (the semantics look very similar to the semantics of first-order logic).

Unification of the syntax of PNL is decidable.

Arithmetic can be finitely axiomatised in PNL and this axiomatisation is, in a sense that can be made formal, correct.

I am confident that the same is true of first-order logic and the lambda-calculus.

In short: PNL and permissive-nominal sets are a formal syntax and semantics that are 'ϵ away' from the informal meta-level.

Some formal syntax

Fix two disjoint countably infinite sets of atoms $\mathbb{A}^<$ and $\mathbb{A}^>$. Write $\mathbb{A} = \mathbb{A}^< \uplus \mathbb{A}^>$.

Let a, b, c range over distinct atoms (the permutative convention).

A permission set has the form $(\mathbb{A}^< \setminus A) \cup B$ where $A \subseteq \mathbb{A}^<$ and $B \subseteq \mathbb{A}^>$ are finite.

An example permission set is $(\mathbb{A}^< \setminus \{a\}) \cup \{b, c\}$.

For each permission set fix a disjoint countably infinite set of unknowns X, Y, Z. Write $p(X)$ for the permission set of X.

Think of $p(X)$ as a sort or type of X.

Some formal syntax

A permutation π is a bijection on atoms such that $\mathit{nontriv}(\pi) = \{a \in \mathbb{A} \mid \pi(a) \neq a\}$ is finite.

Think of π as an α-renaming; we can α-rename finitely many atoms at a time.

Terms are inductively defined by:

$$r ::= a \mid \pi{\cdot}X \mid \mathsf{f}(\mathsf{r}, \ldots, \mathsf{r}) \mid [\mathsf{a}]\mathsf{r}$$

a is like 'x'; X is like 't'; f is like 'λ', '$\forall$', '$|$', or application. $[a]r$ is an atoms-abstraction; it is like the '$x.t$' or '$x.\psi$' in '$\lambda x.t$' or '$\forall x.\psi$'.

Predicates are inductively defined by

$$\phi ::= \bot \mid \phi \Rightarrow \phi \mid \mathsf{P}(\mathsf{r}, \ldots, \mathsf{r}) \mid \forall \mathsf{X}.\phi$$

as in first-order logic.

Free atoms

Things start to get fun when you define free atoms, the permutation action, and α-equivalence.

Define free atoms $fa(r)$ by:

$$fa(\pi{\cdot}X) = \{\pi(a) \mid a \in p(X)\} \quad fa([a]r) = fa(r) \setminus \{a\}$$
$$fa(\mathsf{f}(\mathsf{r}_1, \ldots, \mathsf{r}_\mathsf{n})) = \bigcup fa(r_i) \qquad fa(a) = \{a\}$$

$[a]r$ binds a in r.

$\pi{\cdot}X$ has an infinite set of free atoms $\pi(a), \pi(b), \pi(c)$; this reflects the fact that the informal meta-variable 't' means 'any term' and so could evaluate to any x, y, or z.

Free unknowns $fV(r)$ and $fV(\phi)$ is standard. $X \in fV(\pi{\cdot}X)$ and $X \notin fV(\forall X.\phi)$.

Permutations and α-equivalence

$$\pi{\cdot}a \equiv \pi(a) \qquad \pi{\cdot}\mathsf{f}(\mathsf{r_1},\ldots,\mathsf{r_n}) \equiv \mathsf{f}(\pi{\cdot}\mathsf{r_1},\ldots,\pi{\cdot}\mathsf{r_n})$$

$$\pi{\cdot}[a]r \equiv [\pi(a)]\pi{\cdot}r \qquad \pi{\cdot}(\pi'{\cdot}X) \equiv (\pi\circ\pi'){\cdot}X$$

$$\pi{\cdot}\bot \equiv \bot \qquad \pi{\cdot}(\phi \Rightarrow \psi) \equiv (\pi{\cdot}\phi) \Rightarrow (\pi{\cdot}\psi)$$

$$\pi{\cdot}\mathsf{P}(\mathsf{r_1},\ldots,\mathsf{r_n}) \equiv \mathsf{P}(\pi{\cdot}\mathsf{r_1},\ldots,\pi{\cdot}\mathsf{r_n}) \qquad \pi{\cdot}(\forall X.\phi) \equiv \forall X.\pi{\cdot}\phi$$

Write $(b\ a)$ for the swapping mapping a to b, b to a, and c to c.

α-equivalence is the least congruence such that:

$$\frac{(b\ a){\cdot}r =_\alpha s \quad (b \notin fa(r))}{[a]r =_\alpha [b]s} \qquad \frac{(\pi(a) = \pi'(a) \text{ all } a \in p(X))}{\pi{\cdot}X =_\alpha \pi'{\cdot}X}$$

$$\frac{(Y\ X){\cdot}\phi =_\alpha \psi \quad (Y \notin fV(\phi))}{\forall X.\phi =_\alpha \forall Y.\psi}$$

Some examples of α-equivalence

It is all in the following examples:

$$\begin{array}{rcll} \forall X.X{=}X & =_\alpha & \forall Y.Y{=}Y & \text{if } p(X) = p(Y) \\ [a]a & =_\alpha & [b]b & \\ \forall X.\nu([a]X){=}X & =_\alpha & \forall X.\nu([b](b\ a)\cdot X){=}X & \text{if } b \notin p(X) \end{array}$$

Unknowns can be α-renamed as usual. Atoms can be α-renamed fresh but the α-renaming suspends on unknowns, as a permutation.

The derivation rules

$$\frac{}{\Phi,\ \phi \vdash \phi,\ \Psi}\ (\mathbf{Ax}) \qquad \frac{}{\Phi,\ \bot \vdash \Psi}\ (\bot\mathbf{L})$$

$$\frac{\Phi \vdash \phi,\ \Psi \quad \Phi,\ \psi \vdash \Psi}{\Phi,\ \phi \Rightarrow \psi \vdash \Psi}\ (\Rightarrow\mathbf{L}) \qquad \frac{\Phi,\ \phi \vdash \psi,\ \Psi}{\Phi \vdash \phi \Rightarrow \psi,\ \Psi}\ (\Rightarrow\mathbf{R})$$

$$\frac{\Phi,\ \phi \vdash \Psi}{\Phi,\ \pi{\cdot}\phi \vdash \Psi}\ (\mathit{И})$$

$$\frac{\Phi,\ \phi[X{:=}r] \vdash \Psi \quad (\mathit{fa}(r) \subseteq p(X),\ r{:}s(X))}{\Phi,\ \forall X.\phi \vdash \Psi}\ (\forall\mathbf{L}) \qquad \frac{\Phi \vdash \phi,\ \Psi \quad (X \notin \mathit{fV}(\Phi, \Psi))}{\Phi \vdash \forall X.\phi,\ \Psi}\ (\forall\mathbf{R})$$

$$\frac{\Phi,\ \phi \vdash \Psi \quad (\phi =_\alpha \psi)}{\Phi,\ \psi \vdash \Psi}\ (\alpha_{\mathbf{L}}) \qquad \frac{\Phi \vdash \phi,\ \Psi \quad (\phi =_\alpha \psi)}{\Phi \vdash \psi,\ \Psi}\ (\alpha_{\mathbf{R}})$$

The derivation rules

The $fa(r) \subseteq p(X)$ in ($\forall$**L**) means "$\phi[X{:=}r]$ for every r such that $fa(r) \subseteq p(X)$".

So $\forall X.\mathrm{P(X)} \vdash \mathrm{P(b)}$ for all $b \in p(X)$.

This restriction is not all it seems.

By considering the swapping $(a\ b)$ and (И), $\forall X.\mathrm{P(X)} \vdash \mathrm{P(a)}$ for all a, even if $a \notin p(X)$.

$\forall X.\phi$ does not mean "$\phi[X{:=}r]$ for every r", because permutations are bijective. Suppose $a \notin p(X)$. Then $\forall X.\mathrm{P(a,X)} \vdash \mathrm{P(a,b)}$ for all b other than a; no permutation can identify a with some atom $b \in p(X)$.

Axioms for substitution as a PNL theory

$$
\begin{array}{llll}
(\mathbf{sub}\mathrm{var}) & \forall X.\ \mathrm{var(a)[a{\mapsto}X]} & \approx X & \\
(\mathbf{sub}\#) & \forall X, Z.\ Z[a{\mapsto}X] & \approx Z & \\
 & & (p(Z) = (b\ a)\cdot\mathbb{A}^{<}) & \\
(\mathbf{sub}\mathrm{succ}) & \forall X', X.\ \mathrm{succ(X')[a{\mapsto}X]} & \approx \mathrm{succ(X'[a{\mapsto}X])} & \\
(\mathbf{sub}\mathit{op}) & \forall X'', X', X.\ (X''\ op\ X')[a{\mapsto}X] & \approx (X''[a{\mapsto}X]\ op\ X'[a{\mapsto}X]) & \\
 & & (op \in \{+, *, \dot{\Rightarrow}, \dot{\approx}\}) & \\
(\mathbf{sub}\dot{\forall}) & \forall X, Z.\ (\dot{\forall}([b]Z))[a{\mapsto}X] & \approx \dot{\forall}([b](Z[a{\mapsto}X])) & \\
(\mathbf{subid}) & \forall X.\ X[a{\mapsto}\mathrm{var(a)}] & \approx X &
\end{array}
$$

$a \in \mathbb{A}^{<}$ and $b \notin \mathbb{A}^{<}$. The permission set of X'', X', and X is equal to $\mathbb{A}^{<}$. The permission set of Z is equal to $(b\ a)\cdot\mathbb{A}^{<}$.

Axioms for first-order logic as a PNL theory

$$
\begin{array}{lll}
(\dot{\Rightarrow}) & \forall Z', Z.\ \epsilon(Z' \mathbin{\dot{\Rightarrow}} Z) & \Leftrightarrow (\epsilon(Z') \Rightarrow \epsilon(Z)) \\
(\dot{\forall}) & \forall Z.\ \big(\epsilon(\dot{\forall}([a]Z)) & \Leftrightarrow \forall X.\epsilon(Z[a \mapsto X])\big) \\
(\dot{\bot}) & \epsilon(\dot{\bot}) & \Rightarrow \bot \\
(\dot{\approx}) & \forall X', X.\ X' \approx X & \Rightarrow \epsilon(X' \mathbin{\dot{\approx}} X)
\end{array}
$$

Here Z' and Z have sort o and permission set $\mathbb{A}^{<}$; X' and X have sort ι and permission set $\mathbb{A}^{<}$; and $a \in \mathbb{A}^{<}$.

Axioms for arithmetic as a PNL theory

$$\begin{array}{ll}
(\mathbf{PS0}) & \forall X.\ \mathsf{succ}(\mathsf{X}) \approx 0 \Rightarrow \bot \\
(\mathbf{PSS}) & \forall X', X.\ \mathsf{succ}(\mathsf{X}') \approx \mathsf{succ}(\mathsf{X}) \Rightarrow \mathsf{X}' \approx \mathsf{X} \\
(\mathbf{P{+}0}) & \forall X.\ X + 0 \approx X \\
(\mathbf{P{+}succ}) & \forall X', X.\ X' + \mathsf{succ}(\mathsf{X}) \approx \mathsf{succ}(\mathsf{X}') + \mathsf{X} \\
(\mathbf{P{*}0}) & \forall X.\ X * 0 \approx 0 \\
(\mathbf{P{*}succ}) & \forall X', X.\ X' * \mathsf{succ}(\mathsf{X}) \approx (\mathsf{X}' * \mathsf{X}) + \mathsf{X} \\
(\mathbf{PInd}) & \forall Z.\ (\epsilon(Z[a{\mapsto}0]) \Rightarrow \\
& \quad \big(\forall X.(\epsilon(Z[a{\mapsto}X]) \Rightarrow \epsilon(Z[a{\mapsto}\mathsf{succ}(\mathsf{X})]))\big) \Rightarrow \\
& \quad \forall X.\epsilon(Z[a{\mapsto}X]))
\end{array}$$

All variables have permission set $\mathbb{A}^<$, and $a \in \mathbb{A}^<$.

Conclusions

PNL is not obvious, but in a good way; it is not obvious because it captures something important and non-trivial about mathematical reasoning.

First-order logic, which PNL closely resembles in both its syntax and semantics, is not obvious either.

PNL is really good at expressing specifications with binding.

I have not told you about: sorts, semantics, soundness and completeness, proof-theory, cut-elimination, or the proof of correctness for the axiomatisation of arithmetic.

Conclusions

Let me leave you with this thought:

Just as mathematical discourse can be formalised in first-order and higher-order logic, and this is implemented in theorem-provers and programming languages, so it could also be formalised in PNL.

The advantages of doing so are fully-formal reasoning in a language which accurately reflects what we do in informal mathematical practice.

More stuff . . .

The И quantifier

Nominal sets have the new quantifier meaning 'for some/any fresh atom'. Here is an example of something provable in a logic of nominal sets, such as nominal logic or FM sets:

$$\forall x.(P(x) \Rightarrow \text{И}a.Q(a,x)) \quad \Leftrightarrow \quad \forall x.\text{И}a.(P(x) \Rightarrow Q(a,x))$$

Here is the same example rendered in PNL, where $a \notin p(X)$:

$$\forall X.(\mathsf{P}(\mathsf{X}) \Rightarrow \mathsf{Q}(\mathsf{a},\mathsf{X})) \quad \Leftrightarrow \quad \forall \mathsf{X}.(\mathsf{P}(\mathsf{X}) \Rightarrow \mathsf{Q}(\mathsf{a},\mathsf{X}))$$

Another example:

$$\forall x.\text{И}a.\neg P(a,x) \quad \Leftrightarrow \quad \forall x.\neg \text{И}a.P(a,x)$$

is rendered as

$$\forall X.\neg \mathsf{P}(\mathsf{a},\mathsf{x}) \quad \Leftrightarrow \quad \forall \mathsf{X}.\neg \mathsf{P}(\mathsf{X}).$$

Timeline:

Fraenkel-Mostowski/Nominal sets (journal paper newaas-jv 2002).

Nominal terms (two-level language; journal paper nomu-jv 2004).

Nominal algebra (logic over nominal terms with semantics in nominal sets; journal paper nomuae 2009).

Permissive-nominal terms (made possible $\forall X$ and quotient by α-equivalence; journal paper perntu-jv 2010).

Permissive-nominal logic (PPDP 2010).

What, I forgot substitution?

A (level 2) substitution is a map θ from unknowns to terms such that $fa(\theta(X)) \subseteq p(X)$ for all X.

$$
\begin{aligned}
a\theta &\equiv a & \mathsf{f}(\mathsf{r}_1,\ldots,\mathsf{r}_\mathsf{n})\theta &\equiv \mathsf{f}(\mathsf{r}_1\theta,\ldots,\mathsf{r}_\mathsf{n}\theta) \\
([a]r)\theta &\equiv [a](r\theta) & (\pi\cdot X)\theta &\equiv \pi\cdot\theta(X) \\
\bot\theta &\equiv \bot & (\phi \Rightarrow \psi)\theta &\equiv (\phi\theta) \Rightarrow \psi\theta \\
(\mathsf{P}(\mathsf{r}_1,\ldots,\mathsf{r}_\mathsf{n}))\theta &\equiv \mathsf{P}(\mathsf{r}_1\theta,\ldots,\mathsf{r}_\mathsf{n}\theta) & (\forall X.\phi)\theta &\equiv \forall Y.((\,(Y\ X)\cdot\phi)\theta)
\end{aligned}
$$

In the clause for $\forall X$ we rename X to be fresh for $nontriv(\theta)$, if necessary, using a fixed but arbitrary choice of fresh Y for each X, ϕ, θ.