On proof theory

Murdoch J. Gabbay and Claus-Peter Wirth

Thanks to Bob Atkey

Feburary 10, 2012

On proof search

This talk is introducing a paper with Claus-Peter Wirth on the semantics of proof-search.

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 susceptible to automation. Tableau systems are designed with this in mind: to be robust, compact, and efficient.

Tableaux were new to me, so let's take a look at some tableau rules.

Formula ϕ is $\bot$, $\phi \wedge \phi$, $\neg\phi$, and $\forall a.\phi$. A hypersequent $\mathcal{H}$ is $\bigwedge\bigvee\phi_i$ (conjunction of disjunction).

Tableau rules

$$\frac{(\neg\neg\phi \vee \Phi) \wedge \mathcal{H}}{(\phi \vee \Phi) \wedge \mathcal{H}}\,(\alpha\neg\neg) \qquad \frac{(\top \vee \Phi) \wedge \mathcal{H}}{\mathcal{H}}\,(\alpha\top)$$

$$\frac{(\phi \vee \neg\phi \vee \Phi) \wedge \mathcal{H}}{\mathcal{H}}\,(\alpha\mathbf{EM})$$

$$\frac{(\neg(\phi' \wedge \phi) \vee \Phi) \wedge \mathcal{H}}{(\neg\phi' \vee \neg\phi \vee \Phi) \wedge \mathcal{H}}\,(\alpha\neg\wedge) \qquad \frac{((\phi' \wedge \phi) \vee \Phi) \wedge \mathcal{H}}{(\phi' \vee \Phi) \wedge (\phi \vee \Phi) \wedge \mathcal{H}}\,(\beta\wedge)$$

$$\frac{(\neg\forall a.\phi \vee \Phi) \wedge \mathcal{H}}{(\neg\phi[a\mapsto r] \vee \neg\forall a.\phi \vee \Phi) \wedge \mathcal{H}}\,(\gamma\neg\forall) \qquad \frac{(\forall a.\phi \vee \Phi) \wedge \mathcal{H} \quad (a \notin fa(\Phi))}{(\phi \vee \Phi) \wedge \mathcal{H}}\,(\delta^{-}\forall)$$

Tableau rules

These rules reformulae the usual sequent rules.

(α**EM**) corresponds to the axiom rule. ($\gamma^- \neg \forall$) is $\forall$-left introduction. ($\delta^- \neg \forall$) is $\forall$-right introduction.

It is a fact that these rules can be very inefficient in automation.

A number of so-called liberalised δ-rules have been developed, with the goal of being more efficient. Here is a list, just to give you a flavour:

δ^+, δ^{+^+}, δ^*, δ^{*^*}, and δ^ϵ.

Speedups can be significant. (δ^+) allows at least exponential speedup relative to (δ^-) and δ^{+^+} has another exponential speedup relative to (δ^+).

Liberalised δ-rules

I won't show you the rules directly. They can be quite complicated. We'll see a simple abstraction of them later.

The question I asked with Claus-Peter—who is an authority on these rules—is what underlying semantic simplicity we could find that these rules all have in common.

When we see half-a-dozen variations of something, we ask: What underlying entity are these all variations of?

Semantics too weak

We can't directly answer this question using standard maths because the semantics of logic are not rich enough to directly represent what the liberalised δ-rules do.

We need a richer semantics. We also need a richer syntax.

So I'm going to whip through the syntax and semantics of our new paper.

Permissive-nominal terms

Fix atoms a and unknowns X. Think of atoms as variables naming lemmas, and unknowns as variables for proof-search.

Usual sequents (tableau presentation was earlier) do not separate these two classes.

Every unknown X has a permission set $pms(X)$ which is a set of atoms. A permutation π is a finite bijection on atoms.

We have types $\alpha ::= o \mid \iota \mid \alpha \to \alpha$ and terms $r ::= a \mid \pi{\cdot}X \mid \mathsf{f} \mid rr$.

Examples of f are $\wedge$, $\neg$, and $\forall$. A type system assigns terms to types in the usual way.

Unknowns

If $pms(X) = \{a_1, \ldots, a_n\}$ then the unknown X behaves like a Skolemised/raised term $fa_1 \ldots a_n$. Instantiation of X is capturing:

$(\lambda a_1.X)[X{\mapsto}a_1] = \lambda a_1.a_1$, like
$(\lambda a_1, \ldots, a_n.fa_1 \ldots a_n)[f{\mapsto}\lambda a_1, \ldots, a_n.a_1] = \lambda a_1.a_1$.

With nominal terms:

- We don't need higher orders. $\lambda a_1.X$ and $[X{\mapsto}a_1]$ are easier to work with than $\lambda a_1, \ldots, a_n.fa_1 \ldots a_n$ and $[f{\mapsto}\lambda a_1, \ldots, a_n.a_1]$.
- Atoms are symmetric. Symmetry is key.

Nominal terms are symmetric whereas Skolem terms can not be.
Functional arguments must occur in order; Skolem terms spread across many types.

Example: theory of α-equivalence

$\lambda a.X = \lambda b.(b\ a)\cdot X$ if $b \not\in pms(X)$.

α-equivalence can be defined as the least congruence such that $a, b \not\in fa(r)$ implies $(b\ a)\cdot r =_\alpha r$.

Nominal sets

A nominal set is a set with a symmetry action on it by atoms-permutations.

We can use this to interpret logic—including substitution and universal quantification—in a symmetric abstract domain that generalises syntax. Name-symmetry in syntax is directly interpreted in the semantics.

For example nominal sets have a notion of support supp(x) which generalises 'free atoms/variables of'. Open predicates and terms get interpreted as open elements in a nominal set.

Lemma names a are interpreted as nominal atoms.

Nominal sets

An interpretation $\mathcal{I}$ is an assignment to each type α of a nominal set $[\![\alpha]\!]^{\mathcal{I}}$ together with the following data:

1. For each atom $a \in \mathbb{A}_\alpha$ and constant $\mathsf{f} : \alpha$ elements $a^{\mathcal{I}} \in [\![\alpha]\!]^{\mathcal{I}}$ and $\mathsf{f}^{\mathcal{I}} \in [\![\alpha]\!]^{\mathcal{I}}$.
2. For each $x \in [\![\beta]\!]^{\mathcal{I}}$ and $a \in \mathbb{A}_\alpha$, an element $[a]x \in [\![\alpha\to\beta]\!]^{\mathcal{I}}$ such that $a \notin \mathrm{supp}([a]x)$.
3. For each $x \in [\![\alpha\to\beta]\!]^{\mathcal{I}}$ and $y \in [\![\alpha]\!]^{\mathcal{I}}$, an element $x \bullet y \in [\![\beta]\!]^{\mathcal{I}}$.
4. A preorder $\lessapprox$ on $[\![o]\!]^{\mathcal{I}}$.

Model

Call an interpretation $\mathcal{I}$ a model when:

$$
\begin{array}{llrcl}
(\textbf{moda}) & & a^{\mathcal{I}}[a\mapsto x] & = & x \\
(\textbf{mod}\#) & a \notin \mathrm{supp}(z) \Rightarrow & z[a\mapsto x] & = & z \\
(\textbf{modapp}) & & (z' \bullet z)[a\mapsto x] & = & (z'[a\mapsto x]) \bullet (z[a\mapsto x]) \\
(\textbf{mod}[]) & c \notin \mathrm{supp}(x) \Rightarrow & ([c]z)[a\mapsto x] & = & [c](z[a\mapsto x]) \\
(\textbf{modid}) & & z[a\mapsto a^{\mathcal{I}}] & = & z \\
(\textbf{mod}\eta) & a \notin \mathrm{supp}(z) \Rightarrow & [a](z \bullet a^{\mathcal{I}}) & = & z \\
 & & & & \\
(\textbf{Commute}) & & x \wedge y & \approx & y \wedge x \\
(\textbf{Assoc}) & & (x \wedge y) \wedge z & \approx & x \wedge (y \wedge z) \\
(\textbf{Huntington}) & & x & \approx & \neg(\neg x \wedge \neg y) \wedge \neg(\neg x \wedge y) \\
 & & & & \\
(\forall\textbf{E}) & & \forall a.x & \lessapprox & x[a\mapsto u] \\
(\forall\wedge) & & \forall a.(x \wedge y) & \approx & (\forall a.x) \wedge (\forall a.y) \\
(\forall\vee) & a \notin \mathrm{supp}(y) \Rightarrow & \forall a.(x \vee y) & \approx & (\forall a.x) \vee y
\end{array}
$$

What is important about a model?

The axioms resemble a Hilbert axiomatisation but refer to semantics, not syntax; to truth-values, not predicates.

In a moment we'll take advantage of that to interpret liberalised δ-rules.

In "Stone duality for first-order logic" we build a model of these axioms that is topological has nothing to do with syntax. This 'symmetric' account of logic and computational is not circular: it has natural models that are not equal to a Lindenbaum algebra.

Semantics

A valuation maps $X : \alpha$ to $\zeta(X) \in [\![\alpha]\!]^{\mathcal{I}}$. X is open; the capturing instantiation of X translates to a possibly open element for $\zeta(X)$. That is, ζ may map atoms in $pms(X)$ to atoms in supp(ζ(X)).

Define $[\![r]\!]^{\mathcal{I}}_{\zeta} \in [\![\alpha]\!]^{\mathcal{I}}$ by:

$$\begin{array}{ccc} [\![a]\!]^{\mathcal{I}}_{\zeta} = a^{\mathcal{I}} & [\![\mathsf{f}]\!]^{\mathcal{I}}_{\zeta} = \mathsf{f}^{\mathcal{I}} & [\![\pi{\cdot}X]\!]^{\mathcal{I}}_{\zeta} = \pi{\cdot}\zeta(X) \\ [\![rs]\!]^{\mathcal{I}}_{\zeta} = [\![r]\!]^{\mathcal{I}}_{\zeta} \bullet [\![s]\!]^{\mathcal{I}}_{\zeta} & [\![[a]r]\!]^{\mathcal{I}}_{\zeta} = [a][\![r]\!]^{\mathcal{I}}_{\zeta} & \end{array}$$

We can abstract a in $[\![r]\!]^{\mathcal{I}}_{\zeta}$ because open r gets translated to open denotation.

The X are existential since we get to choose ζ.

Liberalised δ-rules: now simple

Call $x \in [\![\alpha]\!]^{\mathcal{I}}$ a minimiser for $[a]z$ when $[a]z \bullet x \approx \forall a.z$.

So x is an element that tries to make $z[a{\mapsto}x]$ false.

Here now is our liberalised δ rule:

$$\frac{(\forall a.\phi \lor \Phi) \land \mathcal{H}}{X{\downarrow}[a]\phi \land (\phi[a{\mapsto}X] \lor \Phi) \land \mathcal{H}}\,(\delta^{\mathsf{x}}\forall)$$

The interpretation of $X{\downarrow}[a]\phi$ is '$\zeta(X)$ minimises $[\![\phi]\!]^{\mathcal{I}}_{\zeta}$'.

If X minimises $[\![\phi]\!]^{\mathcal{I}}_{\zeta}$ then $[\![\phi[a{\mapsto}X]]\!]^{\mathcal{I}}_{\zeta} = [\![\forall a.\phi]\!]^{\mathcal{I}}_{\zeta}$.

What has happened?

Nominal terms do the same job as Skolem functions, but add name-symmetry.

Nominal sets are sets with symmetry. This allows us to give semantics to logic in which names and binding are directly translated into the semantics.

Open terms get translated to open elements. Substitution and quantifiers are translated to semantic maps that satisfy similar properties, on open elements not necessarily of syntax.

It becomes easy to interpret existential variables as used in liberalised δ-rules: they map to open elements of a nominal set, and the maths clicks into place.

Claus-Peter's original paper is 102 pages long.

Take-home slogans

Nominal terms are syntax with name-symmetry. Nominal sets are a semantics with name-symmetry.

These symmetries are invisible if you use functions. This is the key property to give a truly compositional translation of open terms.

To handle open terms and existential variables in proof-search, this is what you need.

Take-home slogans

The implementors got there first. Nominal terms syntax and nominal sets semantics put this in a nice, reusable, simple mathematical box.

Perhaps we can use this simplicity to improve our algorithms; this is Claus-Peter's interest.

It's not just about proof-search. The 'box' can be reused. "Stone duality for first-order logic"; "Nominal Henkin Semantics"; also "One-and-a-halfth order logic".