

On the Expressivity of Minimal Generic Quantification

David Baelde

Parsifal team, INRIA / École Polytechnique

LFMTP 2008

Minimal Generic Quantification

$$\frac{\Gamma, (\sigma, x) \triangleright Px \vdash G}{\Gamma, \sigma \triangleright \nabla x. Px \vdash G} \quad \frac{\Gamma \vdash (\sigma, x) \triangleright Px}{\Gamma \vdash \sigma \triangleright \nabla x. Px}$$

$$\nabla x. ux = vx \equiv u = v \quad \nabla x. \top \equiv \top \quad \nabla x. \perp \equiv \perp$$

$$\nabla x. Px \wedge Qx \equiv (\nabla x. Px) \wedge (\nabla x. Qx) \quad \text{etc.}$$

$$\nabla x. \forall y. Pxy \equiv \forall y'. \nabla x. Px(y'x) \quad \text{etc.}$$

$$\overline{\Gamma, \sigma \triangleright P\sigma \vdash \sigma \triangleright P\sigma}$$

Implications of minimality:

- ▶ $u = v \equiv \nabla x. u = v$
- ▶ $\vdash \nabla^r x. \exists^r y. \top$ but not necessarily $\vdash \exists^r y. \top$
- ▶ $\nabla x. p y \not\equiv p y$ for an undefined atom p

Fixed points (μ LJ)

$$\begin{aligned} \text{nat } z &\stackrel{\Delta}{=} \top \\ \text{nat } (s X) &\stackrel{\Delta}{=} \text{nat } X \end{aligned}$$

$$\text{nat} \stackrel{\text{def}}{=} \mu(\lambda N \lambda x. x = z \vee \exists y. x = (s y) \wedge N y)$$

Rules for building a fixed point and doing a case-analysis on it:

$$\frac{\Gamma, B(\mu B) \vec{t} \vdash G}{\Gamma, \mu B \vec{t} \vdash G} \quad \frac{\Gamma \vdash B(\mu B) \vec{t}}{\Gamma \vdash \mu B \vec{t}}$$

Induction:

$$\frac{\Gamma, S \vec{t} \vdash G \quad B S \vec{x} \vdash S \vec{x}}{\Gamma, \mu B \vec{t} \vdash G}$$

Dually we define **coinductive** judgments.

Fixed points + ∇

$$\frac{\Gamma, \sigma \triangleright B(\mu B)(\overrightarrow{t\sigma}) \vdash G}{\Gamma, \sigma \triangleright \mu B(\overrightarrow{t\sigma}) \vdash G} \quad \frac{\Gamma \vdash \sigma \triangleright B(\mu B)(\overrightarrow{t\sigma})}{\Gamma \vdash \sigma \triangleright \mu B(\overrightarrow{t\sigma})}$$

All is fine with **finite fixed points**: finite π -calculus, Bedwyr, etc.

Fixed points + ∇

$$\frac{\Gamma, \sigma \triangleright B(\mu B)(\overrightarrow{t\sigma}) \vdash G}{\Gamma, \sigma \triangleright \mu B(\overrightarrow{t\sigma}) \vdash G} \quad \frac{\Gamma \vdash \sigma \triangleright B(\mu B)(\overrightarrow{t\sigma})}{\Gamma \vdash \sigma \triangleright \mu B(\overrightarrow{t\sigma})}$$

Less **natural**:

$$\frac{\Gamma, \sigma \triangleright S(\overrightarrow{t\sigma}) \vdash G \quad BS\overrightarrow{X} \vdash S\overrightarrow{X}}{\Gamma, \sigma \triangleright \mu B(\overrightarrow{t\sigma}) \vdash G} \quad \frac{}{\Gamma, \sigma \triangleright \mu B(\overrightarrow{t\sigma}) \vdash \sigma \triangleright \mu B(\overrightarrow{t\sigma})}$$

We have a **real problem**:

$$\nabla x. \text{nat } y \neq \text{nat } y$$

Who's the **culprit**: **axiom** or **induction**?

- ▶ First answer: **the axiom**. Solution: add structural rules, obtain **LG**. Drawback: we did not learn about the problem, and we **lost minimality**.

Concretely, why want minimal ∇ ?

Let's specify provability in some object logic:

$$\text{prove } \Gamma (\hat{\forall}x. P x) \triangleq \nabla x. \text{prove } \Gamma (P x)$$

We want **adequacy**:

$$\Sigma; \Gamma \hat{=} G \quad \sim \quad \sigma \triangleright \text{prove } \Gamma G$$

Hence, generic structural rules mean:

$$\Sigma; \Gamma \hat{=} G \quad \equiv \quad \Sigma, x; \Gamma \hat{=} G$$

That is not always wanted:

- ▶ If the object logic has a **cut-rule**, it supposes that you can eliminate cuts which involve the extra variable.
- ▶ If it has an **existential quantifier**, it supposes an infinity of names inhabiting each type.

Towards μLJ^∇

Our approach:

- ▶ Ignore undefined atoms.
- ▶ Focus on the induction rule, making it more expressive.
- ▶ We won't get as many **identities**,
but we'll obtain **reasonable morphisms**.

An high-level walkthrough

$\forall y. (\forall x. \text{nat } y) \supset \text{nat } y$

Proceed by induction:

$$\frac{x \triangleright S y \vdash \text{nat } y \quad B S y \vdash S y}{x \triangleright \text{nat } y \vdash \text{nat } y}$$

The invariant S should state that y ,
when removed from underneath the **context** x ,
is still a natural number (in the **empty context**).

An high-level walkthrough

$\forall y. (\forall x. \text{nat } y) \supset \text{nat } y$

Proceed by induction:

$$\frac{x \triangleright S y \vdash \text{nat } y \quad B S y \vdash S y}{x \triangleright \text{nat } y \vdash \text{nat } y}$$

The invariant S should state that y , when removed from underneath the **context x** , is still a natural number (in the **empty context**).

- ▶ Move the context / binder on y .

An high-level walkthrough

$\forall y. (\forall x. \text{nat } y) \supset \text{nat } y$

Proceed by induction:

$$\frac{S' (\lambda x.y) \vdash \text{nat } y \quad (x \triangleright B)S'y' \vdash y'}{(x \triangleright \text{nat}) (\lambda x.y) \vdash \text{nat } y}$$

The invariant S' should state that y' , when removed from underneath the **context x** , is still a natural number (in the **empty context**).

- ▶ Move the context / binder on y .

An high-level walkthrough

$\forall y. (\nabla x. \text{nat } y) \supset \text{nat } y$

Proceed by induction:

$$\frac{S' (\lambda x. y) \vdash \text{nat } y \quad (x \triangleright B) S' y' \vdash y'}{(x \triangleright \text{nat}) (\lambda x. y) \vdash \text{nat } y}$$

The invariant S' should state that y' ,
when removed from underneath the **context** x ,
is still a natural number (in the **empty context**).

$$\lambda y'. \forall y. y' = (\lambda x. y) \supset \text{nat } y$$

The premises are easy:

- ▶ $(\lambda x. y) = (\lambda x. z) \vdash \text{nat } y$
- ▶ $(\lambda x. y) = (\lambda x. s(p'x)), (\forall p. p' = (\lambda x. p) \supset \text{nat } p) \vdash \text{nat } y$

∇ is (almost) a syntactic sugar

Formulas are quotiented by a transformation ϕ which eliminates toplevel occurrences of generic quantification.

- ▶ Key: ∇ commutes with μ (i.e. $(x \triangleright nat) \equiv \mu(\dots)$).
- ▶ ∇ is not a logical connective but a defined one.

Canonical rules

Same rules as μLJ , i.e. LJ plus:

$$\frac{\Gamma, S\vec{t} \vdash G \quad BS\vec{x} \vdash S\vec{x}}{\Gamma, \mu B\vec{t} \vdash G} \quad \frac{\Gamma \vdash B(\mu B)\vec{t}}{\Gamma \vdash \mu B\vec{t}}$$

$$\frac{}{\Gamma, \mu B\vec{t} \vdash \mu B\vec{t}}$$

The transformation ϕ : more mobility of binders

$$\sigma \triangleright P\sigma \equiv \phi_\sigma(P\sigma)$$

The transformation **internalizes the old equivalences**:

$$\phi_\sigma^\Gamma(\nabla x. P|\Gamma|\sigma x) \equiv \phi_{\sigma,x}^\Gamma(P|\Gamma|\sigma x)$$

$$\phi_\sigma^\Gamma(u\sigma = v\sigma) \equiv u = v \quad \phi_\sigma^\Gamma(\top) \equiv \top \quad \phi_\sigma^\Gamma(\perp) \equiv \perp$$

$$\phi_\sigma^\Gamma(P|\Gamma|\sigma \wedge Q|\Gamma|\sigma) \equiv \phi_\sigma^\Gamma(P|\Gamma|\sigma) \wedge \phi_\sigma^\Gamma(Q|\Gamma|\sigma) \quad \text{idem for } \vee, \supset$$

$$\phi_\sigma^\Gamma(\forall x. P|\Gamma|\sigma x) \equiv \forall x'. \phi_\sigma^\Gamma(P|\Gamma|\sigma(x'\sigma)) \quad \text{idem for } \exists$$

But also treats **fixed points**:

$$\phi_\sigma^\Gamma(\mu(\lambda p \lambda \vec{x}. B|\Gamma|p\sigma \vec{x})(\overrightarrow{t\sigma})) \equiv \mu(\lambda p' \lambda \vec{x}'. \phi_{\sigma'}^{\Gamma, \langle p, \sigma, p' \rangle}(B|\Gamma|p\sigma(\overrightarrow{x'\sigma'})) \overrightarrow{t})$$

$$\phi_{\sigma\sigma'}^{\Gamma, \langle p, \sigma, p' \rangle}(p(t\sigma\sigma')) \equiv \nabla \sigma'. p'(\lambda \sigma. t\sigma\sigma')$$

Example: *membership*

The judgment $mem\ x\ \Gamma$ expresses that $x \in \Gamma$:

$$mem \stackrel{def}{=} \mu(\lambda M \lambda x \lambda \Gamma. \exists hd \exists tl. \\ \Gamma = (hd :: tl) \wedge (x = hd \vee M\ x\ tl))$$

$$\phi_{(y:\beta)}(mem) \stackrel{def}{=} \mu(\lambda M' \lambda x' \lambda \Gamma'. \exists hd' \exists tl'. \\ \Gamma' = (\lambda \beta y. (hd' y) :: (tl' y)) \wedge (x' = hd' \vee M'\ x'\ tl'))$$

Example: λ -terms

The judgment *term* Γ m expresses that m is an untyped λ -term with free variables in Γ — we omit applications for conciseness:

$$\begin{aligned} \text{term} &\stackrel{\text{def}}{=} \mu(\lambda T \lambda \Gamma \lambda t. \\ &\quad \text{mem } t \ \Gamma \\ \vee &\quad \exists f. t = \text{abs } f \wedge \nabla_{\alpha} x. T (x :: \Gamma) (f \ x)) \end{aligned}$$

$$\begin{aligned} \phi_{(y:\beta)}(\text{term}) &\stackrel{\text{def}}{=} \mu(\lambda T' \lambda \Gamma' \lambda t'. \\ &\quad \phi_{(y:\beta)}(\text{mem}) \ t' \ \Gamma' \\ \vee &\quad \exists f'. t' = (\lambda_{\beta} y. \text{abs } (f' \ y)) \wedge \\ &\quad \nabla_{\alpha} x. T' (\lambda_{\beta} y. x :: (\Gamma' \ y)) (\lambda_{\beta} y. f' \ y \ x)) \end{aligned}$$

Meta-theory

Admissibility of generic exchange

For any formula P , and any two generic contexts σ and σ^* permutations of each other, it is provable that $(\nabla\sigma. P\sigma) \supset (\nabla\sigma^*. P\sigma)$.

Lifting derivations

For any σ , the provability of $\Sigma; \Gamma \vdash P$ implies that of $\phi_\sigma(\Sigma; \Gamma \vdash P)$.

Cut-elimination (conjecture)

It is easy to extend the cut-reductions, but some technical details make termination not trivially the same as usual.

Conservativity & expressivity

Provability without (co)induction is the same in μLJ^{∇_0} and μLJ^{∇} .
With (co)induction, μLJ^{∇} is strictly more expressive than μLJ^{∇_0} .

A more precise analysis of structural rules

Definition (Weakenable and Strengthenable formulas)

The fragments \mathcal{W} and \mathcal{S} are mutually defined by the following grammar, where \mathcal{G} denotes a **guard**:

$$\begin{aligned}\mathcal{W} &::= \mathcal{W} \wedge \mathcal{W} \mid \mathcal{W} \vee \mathcal{W} \mid \top \mid \perp \mid u = v \mid \nabla x. \mathcal{W} \mid \mu \mathcal{W} \mid \nu \mathcal{W} \\ &\quad \mid \mathcal{S} \supset \mathcal{W} \mid \exists x. \mathcal{W} \mid \forall x. \mathcal{G}x \supset \mathcal{W} \\ \mathcal{S} &::= \mathcal{S} \wedge \mathcal{S} \mid \mathcal{S} \vee \mathcal{S} \mid \top \mid \perp \mid u = v \mid \nabla x. \mathcal{S} \mid \mu \mathcal{S} \mid \nu \mathcal{S} \\ &\quad \mid \mathcal{W} \supset \mathcal{S} \mid \forall x. \mathcal{S} \mid \exists x. \mathcal{G}x \wedge \mathcal{S}\end{aligned}$$

A **guard** over x' is a formula G such that for any σ :

$$\forall x'. \phi_{y\sigma}(G\sigma(x'y\sigma)) \supset \exists x. x' = (\lambda y. x) \wedge \phi_{\sigma}(G\sigma(x\sigma))$$

Proposition

For any formula $W \in \mathcal{W}$ it is provable that $W \supset \nabla x. W$.
Conversely, $S \in \mathcal{S}$ implies $\nabla x. S \supset S$.

Natural expressivity

It is important that these results are obtained in a **natural** way. Notably, their instances can actually be rediscovered by a naive ∇ -agnostic (co)inductive proof-search tactic.

Example

The judgment **bind** $\Gamma x t$ expresses that $\langle x, t \rangle \in \Gamma$. All the following theorems are proved automatically in Taci:

- ▶ $\forall \Gamma \forall x \forall t. (\nabla a. \text{bind } \Gamma x t) \supset \text{bind } \Gamma x t$ (S)
- ▶ $\forall \Gamma \forall x \forall t. \text{bind } \Gamma x t \supset \nabla a. \text{bind } \Gamma x t$ (W)
- ▶ $\forall \Gamma \forall x \forall t'. (\nabla a. \text{bind } \Gamma x (t' a)) \supset \exists t. t' = (\lambda a. t) \wedge \text{bind } \Gamma x t$
- ▶ $\forall \Gamma \forall x' \forall t'. (\nabla a. \text{bind } \Gamma (x' a) (t' a)) \supset$
 $\exists x \exists t. x' = (\lambda a. x) \wedge t' = (\lambda a. t) \wedge \text{bind } \Gamma x t$

Beyond structural rules

Consider the following **inductive definition** (*app* omitted):

$$\begin{aligned} \text{eq } \Gamma \ M \ N & \triangleq \langle M, N \rangle \in \Gamma \\ \text{eq } \Gamma \ (\text{abs } M) \ (\text{abs } N) & \triangleq \forall x. \text{eq } (\langle x, x \rangle :: \Gamma) (M \ x) (N \ x) \end{aligned}$$

We prove the following **theorem**:

$$\begin{aligned} \forall \Gamma M N. \quad & (\forall a b. \text{eq } (\langle a, b \rangle :: \Gamma) (M \ a) (N \ b)) \\ \supset \quad & (\forall a. \text{eq } (\langle a, a \rangle :: \Gamma) (M \ a) (N \ a)) \end{aligned}$$

by induction with the **invariant**:

$$\lambda \Gamma'' M'' N''. \forall a. \text{eq } (\Gamma'' \ a \ a) (M'' \ a \ a) (N'' \ a \ a)$$

The invariance premises are:

- ▶ $a, b \triangleright \langle M \ a \ b, N \ a \ b \rangle \in \Gamma \ a \ b \vdash a \triangleright \text{eq } (\Gamma \ a \ a) (M \ a \ a) (N \ a \ a)$
- ▶ $a, c \triangleright \text{eq } (\langle c, c \rangle :: (\Gamma \ a \ a)) (M \ a \ a \ c) (N \ a \ a \ c) \vdash$
 $a \triangleright \text{eq } (\Gamma \ a \ a) (\text{abs } (M \ a \ a)) (\text{abs } (N \ a \ a))$

Conclusion

Contributions

- ▶ Re-design **minimal generic quantification** to make it play well with **(co)induction**, getting the expected expressivity.
- ▶ ∇ is not a logical connective: take it seriously.
- ▶ An implementation in Taci:
 - ▶ Partially automated proof-search.
 - ▶ Several examples including subject reduction and type determinacy for Λ^{\rightarrow} .
 - ▶ Try the β release: <http://slimmer.gforge.inria.fr/tac>

Further work

- ▶ Better presentation, notably to make normalization obvious.
- ▶ Extend to higher-order logic.
- ▶ Relate to other work.
- ▶ Re-use lifting in other logics.