Autosubst: Reasoning with de Bruijn Terms and Parallel Substitutions

https://www.ps.uni-saarland.de/autosubst

Steven Schäfer Tobias Tebbi Gert Smolka

Department of Computer Science
Saarland University, Germany

September 1, 2015

De Bruijn Terms [deBruijn 72]

- Represent terms up to renaming of bound variables.

$$\left.\begin{array}{l} y \vdash \lambda x.\, x\, y\, (\lambda x\, y.\, y\, x) \\[2em] y \vdash \lambda a.\, a\, y\, (\lambda b\, c.\, c\, b) \end{array}\right\} \quad y \vdash \lambda.\, 0\, 1\, (\lambda.\, \lambda.\, 0\, 1)$$

- Example: Untyped λ-calculus

$$s, t ::= n \mid s\, t \mid \lambda.\, s \qquad \text{where } n \in \mathbb{N}$$

Parallel Substitutions

- *Substitutions* $\sigma, \tau, \theta : \mathbb{N} \to \text{term}$.
- *Instantiation* $s[\sigma]$ capture-avoiding substitution application.

$$(0\,(\lambda.\,0\,1\,2))[\lambda.\,0, 0, \ldots] = (\lambda.\,0)\,(\lambda.\,0\,(\lambda.\,0)\,1)$$

- Example: β-reduction.

$$(\lambda.\,s)\,t \rhd s[t, 0, 1, \ldots]$$

In particular:

$$x \vdash (\lambda.\,0\;1)\,(\lambda.\,0) \quad \rhd \quad x \vdash (\lambda.\,0)\;0$$

Parallel Substitutions: Representation & Reasoning

Example: Substitutivity

$$s_1 \rhd s_2 \to s_1[\sigma] \rhd s_2[\sigma]$$

When $s_1 = (\lambda.\, s)\, t$, suffices to show:

$$\begin{tikzcd}
s[t \cdot \mathrm{id}][\sigma] \arrow[dr, "\text{normalize}"'] & = & s[0 \cdot \sigma \circ \uparrow][t[\sigma] \cdot \mathrm{id}] \arrow[dl, "\text{normalize}"] \\
& s[t[\sigma] \cdot \sigma] &
\end{tikzcd}$$

$\Rightarrow$ σ-calculus [Abadi et.al.91]

Autosubst

- Given description of a term type...

```
Inductive term : Type :=
  | Var (x : var)
  | App (s t : term)
  | Lam (s : {bind term}).
```

- ...provide substitution operations and automation support.

Lemma step-subst s t :

$s \triangleright t \to \forall \sigma.\ s[\sigma] \triangleright t[\sigma]$.

Proof.

induction 1; constructor; subst; autosubst.

Qed.

Results

- Proofs with de Bruijn are different from paper proofs.
  - \+ Fewer preconditions

$$s^{x\,y}_{t\,u} = s^{y\,x}_{u\,t^y_u} \qquad \text{if } x \neq y \text{ and } x \notin \mathrm{FV}(u)$$

$$\Rightarrow s[\sigma][\tau] = s[\sigma \circ \tau]$$

  - \+ Useful generalizations.

    $\Rightarrow$ Context Morphisms (later!)
  - – Have to generalize paper proofs.

$$[\![s^x_t]\!]_\rho = [\![s]\!]_{\rho^x_{[\![t]\!]_\rho}} \quad \Rightarrow \quad [\![s[\sigma]]\!]_\rho = [\![s]\!]_{\lambda x.\, [\![\sigma(x)]\!]_\rho}$$

  - – Terms less readable

$$\lambda xy.\, x \Rightarrow \lambda.\, \lambda.\, 1$$

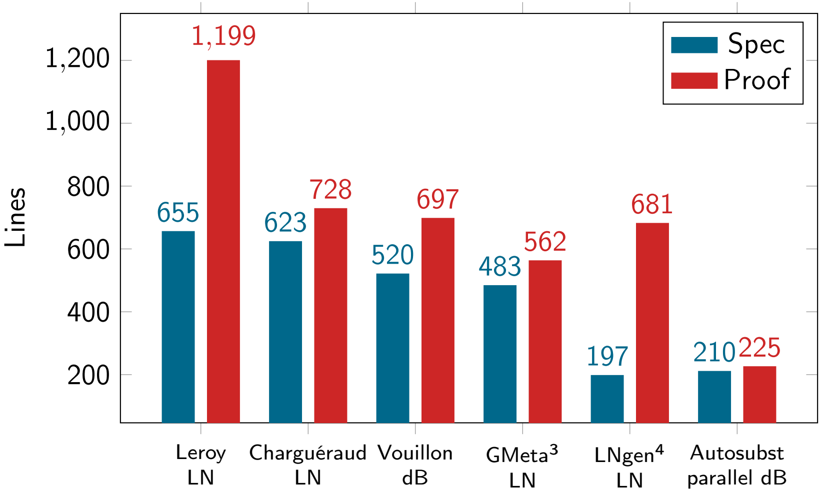
    ...
- Is it worth it?

Case Studies

	Spec[1]	Proof[1]
POPLmark[2]: $F_{<:}$ Preservation & Progress	210	225
$F_{<:}$ Preservation & Progress	185	146
Normalization for CBV System F	99	54
Strong Normalization for System F	153	96
Type Preservation for (predicative) CC_ω	214	229

[1] `coqwc` lines of code
[2] Aydemir et. al. 2005, mostly following the paper proofs

Comparison: POPLmark [Aydemir et. al. 2005]



[3] [Lee, Oliveira, Cho, Yi 2012]

[4] [Aydemir, Weirich 2010]

Substitution Operations (1)

- As in the σ-calculus[5]: $s \cdot \sigma$, $s[\sigma]$, $\sigma \circ \tau$, $\uparrow$.
- Substitutions can be seen as infinite streams
- Cons ($s \cdot \sigma$) adds a new head element

$$\sigma = \begin{cases} 0 \mapsto \sigma(0) \\ 1 \mapsto \sigma(1) \\ 2 \mapsto \sigma(2) \\ 3 \mapsto \sigma(3) \\ \quad\vdots \end{cases} \qquad\qquad s \cdot \sigma = \begin{cases} 0 \mapsto s \\ 1 \mapsto \sigma(0) \\ 2 \mapsto \sigma(1) \\ 3 \mapsto \sigma(2) \\ \quad\vdots \end{cases}$$

[5] [Abadi,Cardelli,Curien,Lévy 91]

Substitution Operations (2)

- Shift ($\uparrow$) increases all free variables by 1

$$\uparrow \; = \begin{cases} 0 \mapsto 1 \\ 1 \mapsto 2 \\ 2 \mapsto 3 \\ \quad \vdots \end{cases}$$

- Identity substitutition $\mathrm{id} = 0 \cdot \uparrow$

Substitution Operations (3)

- Instantiation[6] and composition

$$\begin{aligned} n[\sigma] &= \sigma(n) \\ (s\ t)[\sigma] &= s[\sigma]\ t[\sigma] \\ (\lambda.\ s)[\sigma] &= \lambda.\ s[0 \cdot (\sigma \circ \uparrow)] \end{aligned} \qquad (\sigma \circ \tau)(n) = \sigma(n)[\tau]$$

- Not structurally recursive.
- In a total language, first define instantiation for renamings $\xi : \mathbb{N} \to \mathbb{N}$, then lift to general case [Adams06].

$$(\xi \circ \sigma)(n) = \sigma(\xi(n))$$

[6] [de Bruijn 1972]

Can express β and η-reduction

$$(\lambda.\, s)\, t \rhd s[t \cdot \mathsf{id}] \tag{β}$$

$$\lambda.\, (s[\uparrow]\, 0) \rhd s \tag{η}$$

Substitution Algebra

Axiomatic equality for substitution expressions

$$(s\,t)[\sigma] = s[\sigma]\,t[\sigma] \qquad \mathrm{id} \circ \sigma = \sigma$$

$$(\lambda.\,s)[\sigma] = \lambda.\,s[0 \cdot \sigma \circ \uparrow] \qquad \sigma \circ \mathrm{id} = \sigma$$

$$0[s \cdot \sigma] = s \qquad (\sigma \circ \tau) \circ \theta = \sigma \circ (\tau \circ \theta)$$

$$\uparrow \circ (s \cdot \sigma) = \sigma \qquad (s \cdot \sigma) \circ \tau = s[\tau] \cdot \sigma \circ \tau$$

$$\mathrm{id} := 0 \cdot \uparrow \qquad n := 0[\uparrow^n]$$

- Sound and complete [Schäfer,Smolka,Tebbi 15].
- Deductively equivalent to σ_{SP} [Curien,Hardin,Lévy 96].

$\Rightarrow$ Implement decision procedure with rewriting.

Context Morphisms

- Context morphism lemma: Compatibility of typing with instantiation.

$$\frac{\Gamma \vdash s : A \qquad \sigma : \Delta \to \Gamma}{\Delta \vdash s[\sigma] : A}$$

$$\sigma : \Delta \to \Gamma := \forall x \in \Gamma,\ \Delta \vdash \sigma(x) : \Gamma(x)$$

- A substitution $\sigma : \Delta \to \Gamma$ is called a context morphism.
- The context morphism lemma implies many structural properties of type systems.

Instances of the context morphism lemma

$$\frac{\Gamma \vdash s : A \qquad \sigma : \Delta \to \Gamma}{\Delta \vdash s[\sigma] : A} \qquad\qquad \begin{array}{c}\sigma : \Delta \to \Gamma := \\ \forall x \in \Gamma,\ \Delta \vdash \sigma(x) : \Gamma(x)\end{array}$$

- Structural properties: weakening, substitution, contraction, exchange, ...

$$\begin{aligned} \uparrow &: \Gamma, A \to \Gamma \\ s \cdot \mathrm{id} &: \Gamma \to \Gamma, A \qquad\qquad \text{if } \Gamma \vdash s : A \end{aligned}$$

- Narrowing in $\mathsf{F}_{<:}$

$$\mathrm{id} : \Gamma, A \to \Gamma, B \qquad\qquad \text{if } \Gamma \vdash A <: B$$

- Context conversion in Type Theory

$\vdots$

Substitution Lemmas

$$\frac{\Gamma \vdash s : A \qquad \sigma : \Delta \to \Gamma}{\Delta \vdash s[\sigma] : A}$$

$$\sigma : \Delta \to \Gamma := \\ \forall x \in \Gamma,\ \Delta \vdash \sigma(x) : \Gamma(x)$$

maximal generalization

$$\frac{\Gamma, A, \Delta \vdash s : B \qquad \Gamma \vdash t : A}{\Gamma, \Delta \vdash s[|\Delta| \mapsto t] : B}$$

$$x \mapsto t := 0, 1, .., x - 1, t[\uparrow^x], x, ..$$

minimal generalization

$$\frac{\Gamma, A \vdash s : B \qquad \Gamma \vdash t : A}{\Gamma \vdash s[t \cdot \mathrm{id}] : B}$$

Outlook

- Extension to multi-sorted syntactic theories.
  - Add new substitution operations, e.g. type substitution $s[\![\sigma]\!]$, term substitution $s[\sigma]$), with interactions $s[\sigma][\![\tau]\!] = s[\![\tau]\!][\sigma \bullet \tau]$

    $\Rightarrow$ Details in the paper, implemented in Autosubst.
  - Generalize to vector substitutions, e.g. $s[\sigma, \tau]$, where σ is a term substitution, τ is a type substitution.

    $\Rightarrow$ Future work, currently being implemented.
- Reasoning about free variables.
- Binders with variable arity.
- More robust implementation.

Thank you for your attention!

Try Autosubst today

https://www.ps.uni-saarland.de/autosubst