Autosubst 2: Towards Reasoning with Multi-Sorted de Bruijn Terms and Vector Substitutions

Jonas Kaiser, Steven Schäfer, Kathrin Stark

SAARLAND UNIVERSITY
COMPUTER SCIENCE

September 08, 2017

Our Motivation

- Formalising the metatheory of programming languages and logical systems with binders,
  - e.g. call-by-value System F (F_{CBV}):

$$
\begin{array}{rcll}
A, B \in ty & ::= & X \mid A \to B \mid \forall X.A & \text{Types} \\
s, t \in tm & ::= & s\,t \mid s\,A \mid v & \text{Terms} \\
u, v \in vl & ::= & x \mid \lambda(x : A).s \mid \Lambda X.s & \text{Values}
\end{array}
$$

- Formalising proofs as
  - weak normalisation
  - progress and preservation of type systems

Goal: Weak Normalisation via Logical Relations

Theorem (Weak Normalisation)

$$\vdash s : A \;\rightarrow\; \exists v.\, s \Downarrow v$$

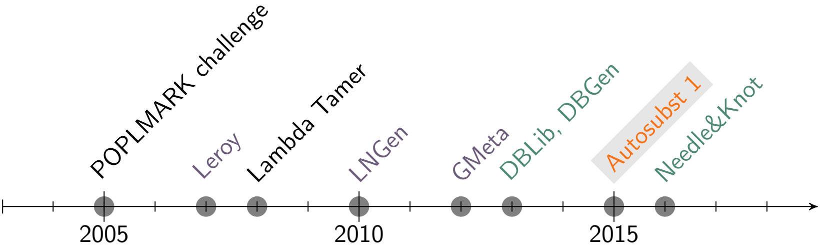
- Substitution and substitution lemmas of the form $s[\sigma] = t[\tau]$ arise everywhere!
  - In the definition of $\vdash s : A$ and $s \Downarrow v$
  - In the definition of term / value interpretations
  - In the proofs that syntactic typing implies semantic typing
- This requires most lines of code:

Related Work

- **Benchmarks:** POPLMARK challenge [Aydemir et al. 2005], POPLMark Reloaded [Abel/Momigliano/Pientka 2017] . . .
- **Representation techniques:** de Bruijn [de Bruijn 1972], locally nameless [Aydemir et al. 2008], nominal logic [Pitts 2001], higher order abstract syntax (HOAS) [Pfenning/Elliot 1988], . . .
- **Proof assistants:** Abella [Baelde et al. 2014], Beluga [Pienta/Cave 2015], . . .

Binders in Coq

- Large user base, mature system
- Dependent types
- No native support for nominal binders/HOAS [Pfenning/Elliot '88]



☐ locally nameless ☐ single-point de Bruijn ☐ parallel de Bruijn

$\sigma = x \mapsto v$ $\sigma = 0 \mapsto v_0, 1 \mapsto v_1, \ldots$

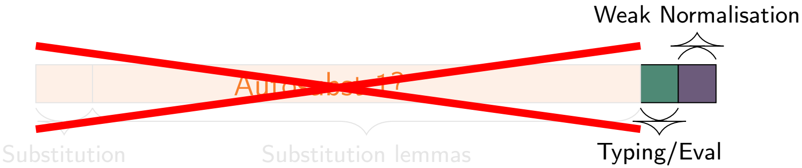
Autosubst 1 [Schäfer/Smolka/Tebbi '15] – A Library à la de Bruijn [de Bruijn '72]

- **Goal:** Given an annotated inductive type, automates the generation of substitution and substitution lemmas
- Variable representation à la [de Bruijn '72]
  $A, B \in ty \ ::= \ X \in \mathbb{N} \mid A \to B \mid \forall.\, A$
- Parallel substitutions $s[\sigma]$ à la [de Bruijn '72]
- Equational theory à la σ-calculus [Abadi et al '91]
  - Substitution is broken down into primitives, e.g. $A \cdot \sigma, \uparrow, \sigma \circ \tau \dots$
  - Decidable, sound, complete rewriting system for UTLC [Schäfer/Smolka/Tebbi '15]

Autosubst 1 [Schäfer/Smolka/Tebbi '15] – A Library à la de Bruijn [de Bruijn '72]

Autosubst 1 was used for:

- Several case studies: Strong normalisation to the metatheory of Martin-Löf type theory [Schäfer/Smolka/Tebbi '15]
- Interactive proofs in higher-order concurrent separation logic [Krebbers et al. '17]
- Equivalence proofs of alternative syntactic presentations of System F [Kaiser et al. '17]
- Formalisations of logical relations for $F\mu$ [Timany et al. '17]
- Formalisation of CPS translations for UTLC [Pottier '17]



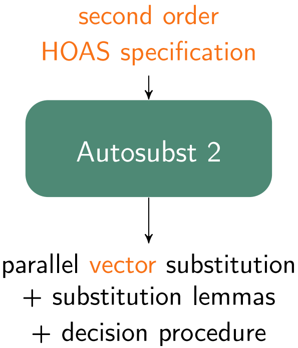
Autosubst 1 Cannot Handle F_{CBV}

$$
\begin{array}{rcll}
A, B \in ty & ::= & \boxed{X} \mid A \to B \mid \forall X.A & \text{Types} \\
s, t \in tm & ::= & s\,t \mid s\,A \mid v & \text{Terms} \\
u, v \in vl & ::= & \boxed{x} \mid \lambda(x : \boxed{A}).\, s \mid \Lambda X.s & \text{Values}
\end{array}
$$

- Enforces variables for each sort with substitutions
- Ad-hoc handling of heterogeneous substitutions
  - Values require type and value variables
  - AS1: One instantiation operation per sort
  - **Problem:** How do they interfere?

$$s[\tau]_{vl}[\sigma]_{ty} = s[\sigma]_{ty}[\lambda x.\, (\sigma\, x)[\tau]_{ty}]_{vl}$$

Contributions of Autosubst 2



- Handle mutually inductive sorts
  1. Extend the input language to second order HOAS
  2. More uniform handling of heterogeneous substitutions
     - Parallelise!

$$s[\sigma_{ty}, \sigma_{vl}]$$

From HOAS to de Bruijn for F_{CBV}

```
ty, tm, vl : Type

arr : ty → ty → ty
all : (ty → ty) → ty


app : tm → tm → tm
tapp: tm → ty → tm
vt  : vl → tm


lam : ty → (vl → tm) → vl
tlam: (ty → tm) → vl
```

```
Inductive ty : Type :=
  | var_ty : index → ty
  | arr : ty  → ty → ty
  | all : ty → ty.

Inductive tm  : Type :=
  | app : tm → tm → tm
  | tapp : tm → ty → tm
  | vt : vl → tm
 with vl  : Type :=
  | var_vl : index → vl
  | lam : ty → tm → vl
  | tlam : tm → vl.
```

1. Which sorts depend on each other?
2. Which sorts require variable constructors?
3. What are the components of the substitution vectors?

Dependency Graph for F_{CBV}

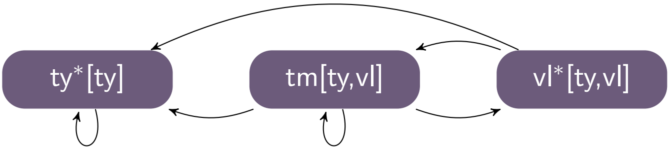
```
ty, tm, vl : Type

arr : ty → ty → ty
all : (ty → ty) → ty

app : tm → tm → tm
tapp: tm → ty → tm
vt  : vl → tm

lam : ty → (vl → tm) → vl
tlam: (ty → tm) → vl
```

1. Which sorts depend on each other?
2. Which sorts require variable constructors (*)?
3. What are the components of the substitution vectors?



Contributions of Autosubst 2

second order
HOAS specification

↓

Autosubst 2

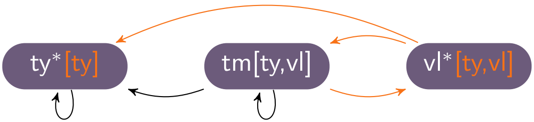
↓

parallel vector substitution
+ substitution lemmas
+ decision procedure

- Handle mutually inductive sorts
  1. Extend the input language to second order HOAS
  2. More uniform handling of heterogeneous substitutions
     - Parallelise!

$$s[\sigma_{ty}, \sigma_{vl}]$$

Towards Vector Substitutions



$$x[\sigma, \tau] = \tau\, x$$
$$(\lambda A.\ s)[\sigma, \tau] = \lambda A[\sigma].\ s[\Uparrow_{tm}^{vl} (\sigma, \tau)]$$
$$(\Lambda.\ s)[\sigma, \tau] = \Lambda.\ s[\Uparrow_{tm}^{ty} (\sigma, \tau)]$$

$$\Uparrow_{tm}^{vl} (\sigma, \tau) = (\sigma, 0_{vl} \cdot \tau \circ (\mathrm{id}_{ty}, \uparrow))$$
$$\Uparrow_{tm}^{ty} (\sigma, \tau) = (0_{ty} \cdot \sigma \circ \uparrow, \tau \circ (\uparrow, \mathrm{id}_{vl}))$$

- Traverses values
  - homomorphically
  - mutually recursive
  - with the inferred vector
- Take care of:
  - Projections
  - Castings
  - Traversals of binders

Towards an Equational Theory of Vector Substitutions

Given Extended (vector) primitives $A \cdot \sigma$, $\sigma \circ (\sigma', \tau')$, ...

Goal Extend the σ-calculus to multi-sorted syntax

Example: Adapt the Equations From Single-sorted to Multi-sorted

1. Defining equations of instantiation
2. Interaction between lift and cons, e.g.

$$\uparrow \circ (s \cdot \sigma) \equiv \sigma$$

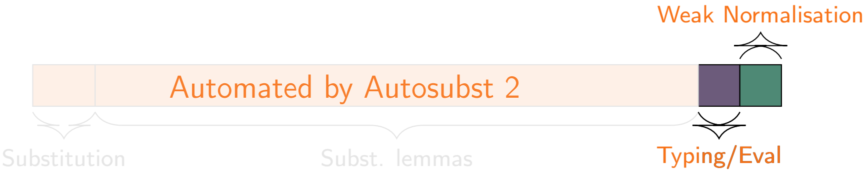
3. Monoid action laws, e.g.

$A[\mathrm{id}_{ty}] = A$	$s[\mathrm{id}_{ty}, \mathrm{id}_{vl}] = s$
$\mathrm{id}_{ty} \circ \sigma \equiv \sigma$	$\mathrm{id}_{ty} \circ (\sigma, \tau) \equiv \sigma$
	$\mathrm{id}_{vl} \circ (\sigma, \tau) \equiv \tau$
$A[\sigma][\sigma'] = A[\sigma \circ \sigma']$	$s[\sigma, \tau][\sigma', \tau'] = s[\sigma \circ \sigma', \tau \circ (\sigma', \tau')]$

Technical Remarks

- Research prototype
- **Input:** Second Order HOAS signature (F_{CBV}: $\sim$ 10 lines)
- **Output**: Coq source file (F_{CBV}: $\sim$ 1600 lines)

1% De Bruijn terms
9% Instantiation
90% Generated substitution lemmas/ automation



Weak Normalisation of $\mathsf{F}_{\mathsf{CBV}}$

- Definition of *typing* $\Gamma \vdash s : A$ and $\Gamma \vdash^{v} v : A$, e.g.

$$\frac{\Gamma \vdash s : \forall . A}{\Gamma \vdash s\, B : A[B \cdot \mathrm{id}_{ty}]}$$

Substitution generated by Autosubst 2

- Definition of *big-step evaluation* $s \Downarrow v$, e.g.

$$\frac{s \Downarrow \lambda A.\, b \qquad t \Downarrow u \qquad b[\mathrm{id}_{ty}, u \cdot \mathrm{id}_{vl}] \Downarrow v}{s\, t \Downarrow v}$$

(40 loc)

Theorem (Weak Normalisation)

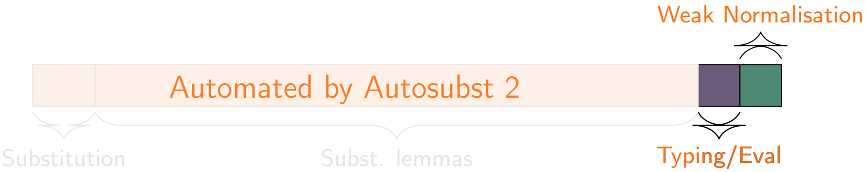
For all s, A we have

$$\vdash s : A \;\rightarrow\; \exists v.\, s \Downarrow v$$

Technical Remarks

- Research prototype
- **Input:** Second Order HOAS signature (F_{CBV}: $\sim$ 10 lines)
- **Output**: Coq source file (F_{CBV}: $\sim$ 1600 lines)

  1% De Bruijn terms
  9% Instantiation
  90% Generated substitution lemmas/ automation



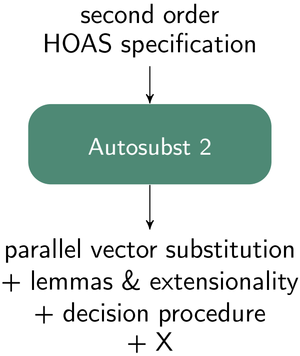
A Formalised Proof of Weak Normalisation

1. Define a term interpretation $[\![A]\!]_\rho$/ value interpretation $(\!|A|\!)_\rho$. (20 loc)
2. Term/value interpretation are compatible with substitution. (30 loc)
3. Define semantic counterparts to the syntactic typing relations, e.g.
   $\Gamma \vDash^{v} v : A := \forall \sigma \tau \rho.\ (\!|\Gamma|\!)_\rho\ \tau \to (\!|A|\!)_\rho\ v[\sigma, \tau]$ (5 loc)
4. Prove that syntactic typing implies semantic typing. (25 loc)
5. Show weak normalisation. (10 loc)

- Substitution and substitution lemmas of the form $s[\sigma] = t[\tau]$ are automatically solved by Autosubst 2
  - for example:

$$s[\Uparrow^{vl}_{tm} (\sigma, \tau)][\mathrm{id}_{ty}, v \cdot \mathrm{id}_{vl}] = s[\sigma, v \cdot \tau]$$

Future Work



1. Testing the current development
   1.1 Prove properties of extended TRS
   1.2 More case studies
2. Efficiency and user interface
   2.1 Plugin in Coq
   2.2 Normalisation procedure
3. Extensions
   3.1 Allow more expressive input languages
   3.2 More proof automation following ideas of [Allais et al. '17]

A Formalised Proof of Weak Normalisation

1. Define a term interpretation $[\![A]\!]_\rho$/ value interpretation $(\!|A|\!)_\rho$. (20 loc)
2. Term/value interpretation are compatible with substitution. (30 loc)
3. Define semantic counterparts to the syntactic typing relations, e.g.
   $\Gamma \vDash^v v : A := \forall \sigma \tau \rho.\ (\!|\Gamma|\!)_\rho\, \tau \to (\!|A|\!)_\rho\ v[\sigma, \tau]$ (5 loc)
4. Prove that syntactic typing implies semantic typing. (25 loc)
5. Show weak normalisation. (10 loc)

- Substitution and substitution lemmas of the form $s[\sigma] = t[\tau]$ are automatically solved by Autosubst 2
  - for example:

$$s[\Uparrow^{vl}_{tm} (\sigma, \tau)][\mathrm{id}_{ty}, v \cdot \mathrm{id}_{vl}] = s[\sigma, v \cdot \tau]$$

Recap and Contributions

- Preliminary version of Autosubst 2 is available
- Extends Autosubst 1 to handle mutually inductive types by using parallel *vector* substitutions
- Extends the equational theory and automatisation of Autosubst 1
- Work in progress – there remains a lot to be done!

www.ps.uni-saarland.de/extras/lfmtp17