Strong Normalization of the λ-calculus in Lean

Bachelor Talk

Sarah Mameche

Advisor: Kathrin Stark
Supervisor: Prof. Gert Smolka

Programming Systems Lab
Saarland University

Feburary 2, 2019

Motivation

Promise: Interactive + automated theorem proving

Our goal: Formalise metatheory of calculi with binders in Lean

Motivation

Our goal: Formalise metatheory of calculi with binders in Lean

- Normalization proofs for $\mathsf{STLC}_{\mathsf{num}}$

$$A, B \in \mathsf{ty} := \mathsf{int} \mid A \to B$$
$$s, t \in \mathsf{tm} := x \mid \lambda x.s \mid s\ t \mid \overline{n} \mid s + t \quad (n \in \mathbb{N})$$

Motivation

Our goal: Formalise metatheory of calculi with binders in Lean

- Normalization proofs for $\mathrm{STLC}_{\mathrm{num}}$

Challenge: Binders and substitution everywhere

- Technical in proof assistants
- POPLMark [Aydemir et al 2005]
- Need tool support , e.g. LNGen [Aydemir et al 2010], Autosubst [Schäfer et al 2015, Stark et al 2018]

How well does Lean adapt?

Contributions

1. Adaptation of Autosubst to Lean
   - Tool for binder support in Lean
   - Automation approaches

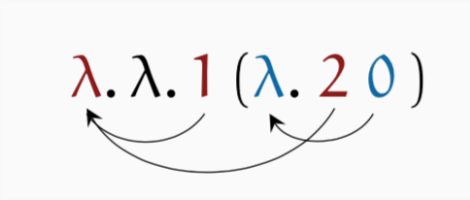
2. Case study
   - Weak normalization of $\mathsf{STLC}_{\mathsf{num}}$
   - Strong normalization of $\mathsf{STLC}_{\mathsf{num}}$

Contributions

1. Adaptation of Autosubst to Lean
   - Tool for binder support in Lean
   - Automation approaches
2. Case study
   - Weak normalization of $\mathrm{STLC}_{\mathrm{num}}$
   - Strong normalization of $\mathrm{STLC}_{\mathrm{num}}$

Reminder: Autosubst [Schäfer et al 2015, Stark et a 2018]

- De Bruijn indices and parallel substitutions [de Bruijn 1972]



$$\sigma, \tau : \mathbb{N} \to \mathrm{tm}$$

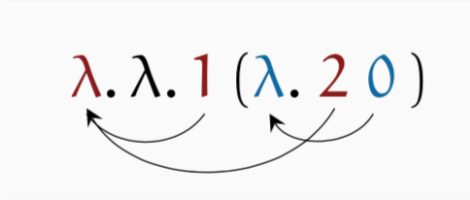
$$A, B \in \mathrm{ty} := \mathrm{int} \mid A \to B$$

$$s, t \in \mathrm{tm} := x \mid \lambda s \mid s\ t \mid \overline{n} \mid s + t \quad (x, n \in \mathbb{N})$$

- Substitution primitives of the σ-calculus [Abadi 1991]
  = equational theory with decidable, sound, and complete rewriting rules

Reminder: Autosubst [Schäfer et al 2015, Stark et a 2018]

- De Bruijn indices and parallel substitutions [de Bruijn 1972]



$$\sigma, \tau : \mathbb{N} \to \mathrm{tm}$$

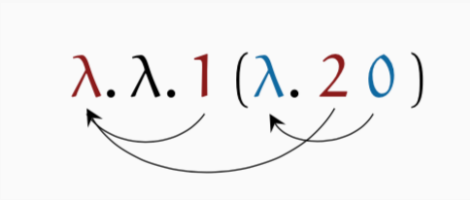
$$A, B \in \mathrm{ty} := \mathrm{int} \mid A \to B$$

$$s, t \in \mathrm{tm} := x \mid \lambda s \mid s\ t \mid \overline{n} \mid s + t \quad (x, n \in \mathbb{N})$$

- Substitution primitives of the σ-calculus [Abadi 1991]
  = equational theory with decidable, sound, and complete rewriting rules

Reminder: Autosubst [Schäfer et al 2015, Stark et a 2018]

- De Bruijn indices and parallel substitutions [de Bruijn 1972]



$$\sigma, \tau : \mathbb{N} \to \mathrm{tm}$$

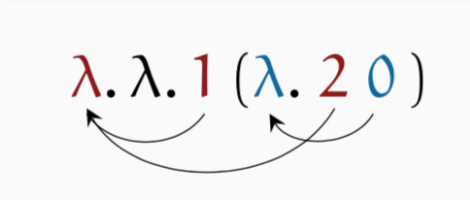
$$A, B \in \mathrm{ty} := \mathrm{int} \mid A \to B$$

$$s, t \in \mathrm{tm} := x \mid \lambda s \mid s\ t \mid \overline{n} \mid s + t \quad (x, n \in \mathbb{N})$$

- Substitution primitives of the σ-calculus [Abadi 1991]
  = equational theory with decidable, sound, and complete rewriting rules

Reminder: Autosubst [Schäfer et al 2015, Stark et a 2018]

- De Bruijn indices and parallel substitutions [de Bruijn 1972]



$$\sigma, \tau : \mathbb{N} \to \mathrm{tm}$$

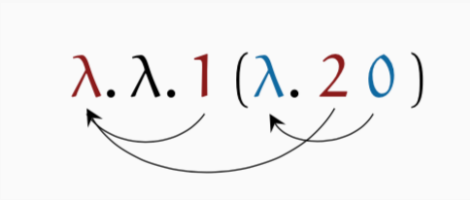
$$A, B \in \mathrm{ty} := \mathrm{int} \mid A \to B$$

$$s, t \in \mathrm{tm} := x \mid \lambda s \mid s\ t \mid \overline{n} \mid s + t \quad (x, n \in \mathbb{N})$$

- Substitution primitives of the σ-calculus [Abadi 1991]
  = equational theory with decidable, sound, and complete rewriting rules

Reminder: Autosubst [Schäfer et al 2015, Stark et a 2018]

- De Bruijn indices and parallel substitutions [de Bruijn 1972]



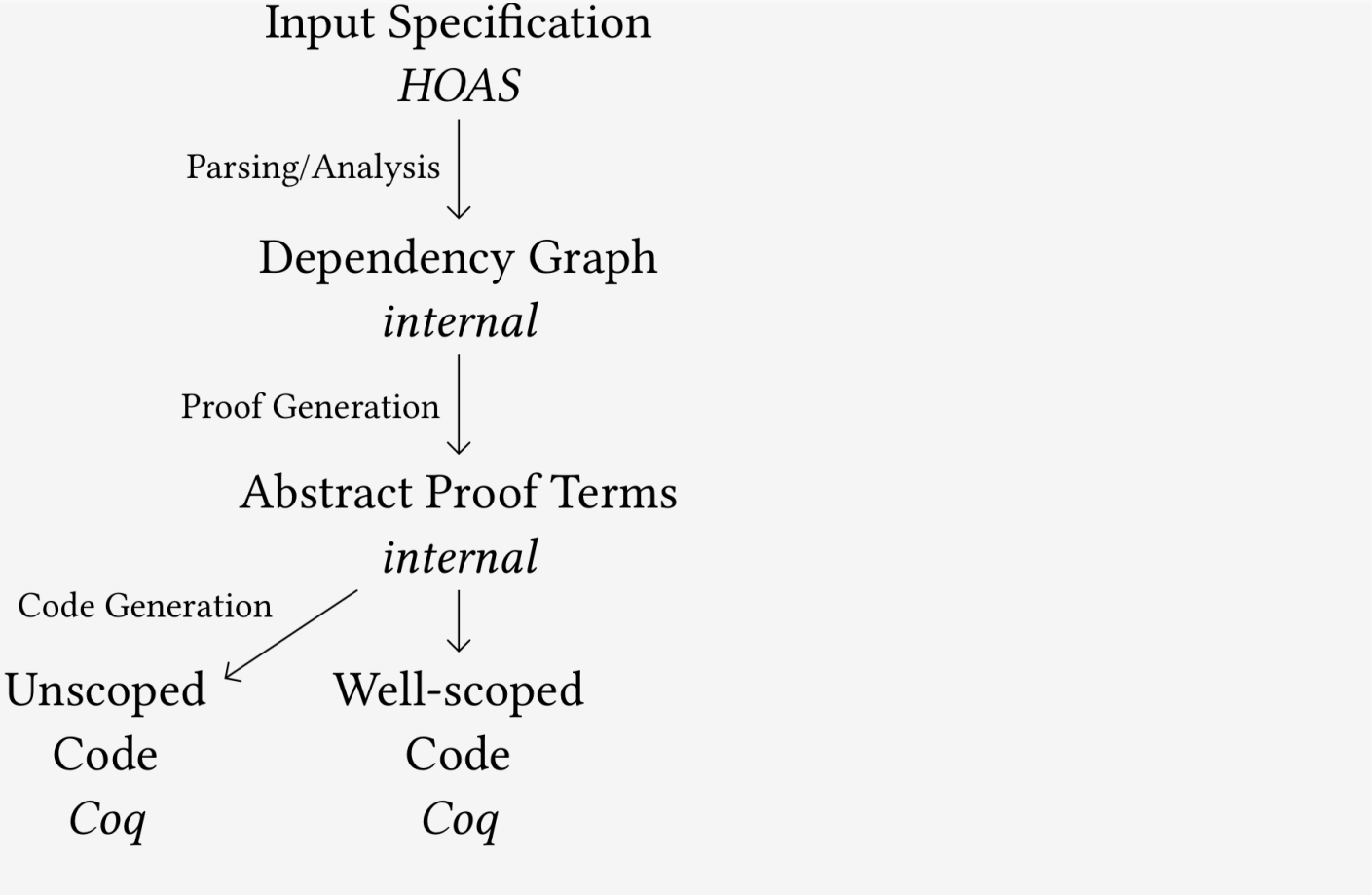
$$\sigma, \tau : \mathbb{N} \to \mathrm{tm}$$

$$A, B \in \mathrm{ty} := \mathrm{int} \mid A \to B$$

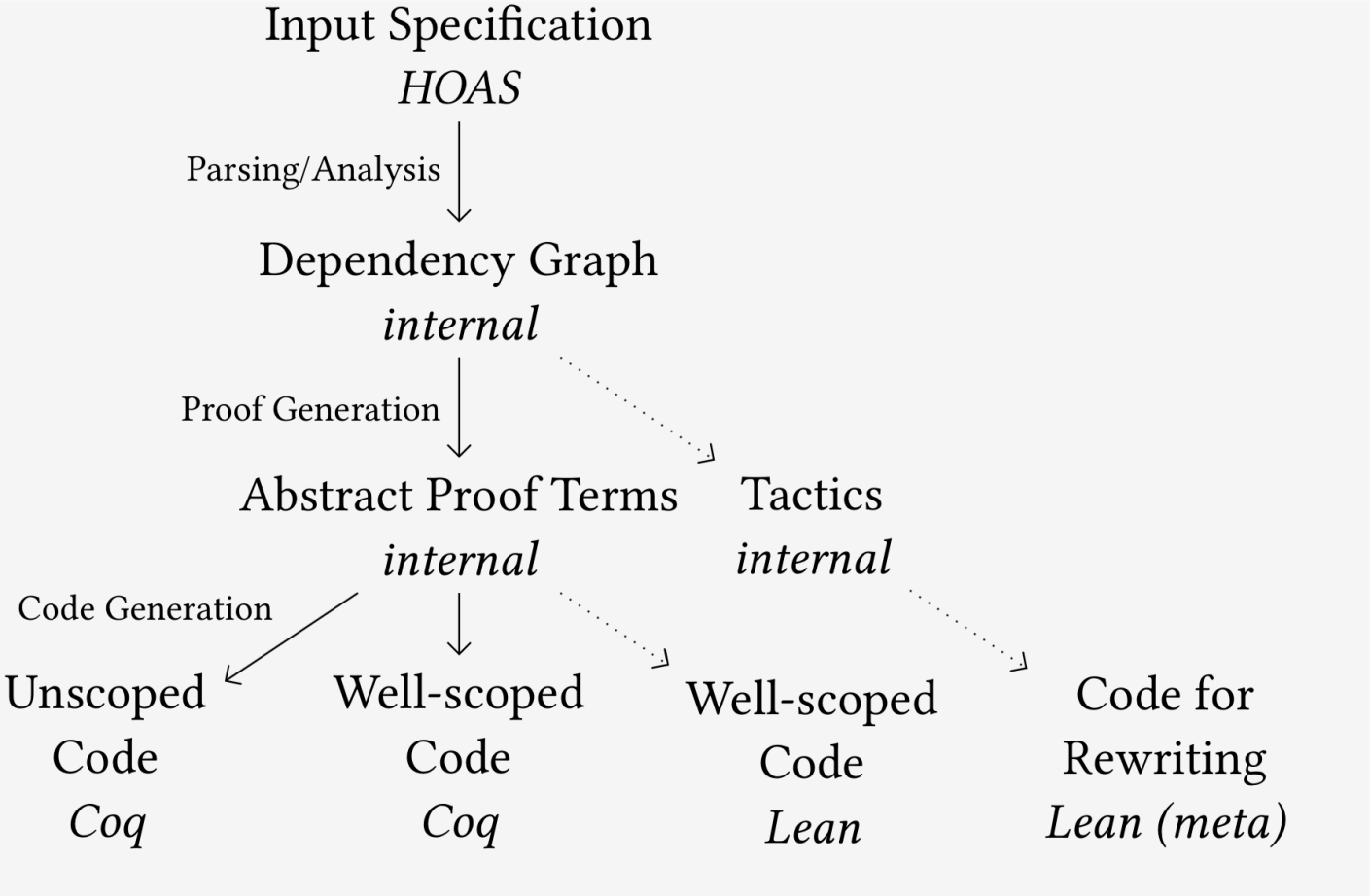
$$s, t \in \mathrm{tm} := x \mid \lambda s \mid s\ t \mid \overline{n} \mid s + t \quad (x, n \in \mathbb{N})$$

- Substitution primitives of the σ-calculus [Abadi 1991]
  = equational theory with decidable, sound, and complete rewriting rules

AS 2 — Components



AS in Lean — Components



AS in Lean — Code Generation

```
tm: Type
nat : Type

app : tm -> tm -> tm
lam : (tm -> tm) -> tm

const : nat -> tm
plus : tm -> tm -> tm
```

→

```
inductive tm  : nat -> Type
  | var_tm : Π {ntm : nat}, Fin ntm -> tm ntm
  | app : Π {ntm : nat}, tm ntm -> tm ntm -> tm ntm
  | lam : Π {ntm : nat}, tm (nat.succ ntm) -> tm ntm
  | const : Π {ntm : nat}, nat  -> tm ntm
  | plus : Π {ntm : nat}, tm ntm -> tm ntm -> tm ntm
open tm


def subst_tm : Π { mtm ntm} (sigmatm : Fin mtm -> tm ntm)
  (s : tm mtm), tm ntm
    | mtm ntm sigmatm  (var_tm s) := sigmatm s
    | mtm ntm sigmatm  (app s0 s1) :=
       app (subst_tm sigmatm s0) (subst_tm sigmatm s1)
    | mtm ntm sigmatm  (lam s0) :=
       lam (subst_tm (up_tm_tm sigmatm) s0)
    | mtm ntm sigmatm  (const s0) := const s0
    | mtm ntm sigmatm  (plus s0 s1) :=
       plus (subst_tm sigmatm s0) (subst_tm sigmatm s1)
```

Reminder: Automation approaches

- Rewriting
- Lean's simplifier
- Expression matching
- Reification

How does it work in practice?

Contributions

1. Adaptation of Autosubst to Lean
   - tool for binder support in Lean
   - automation approaches

2. Case study
   - Weak normalization of $\mathsf{STLC}_{\mathsf{num}}$ $\vdash s : A \rightarrow \exists v. s \succ^* v$
   - Strong normalization of $\mathsf{STLC}_{\mathsf{num}}$ $\vdash s : A \rightarrow \mathrm{SN}\ s$

Contributions

1. Adaptation of Autosubst to Lean
   - tool for binder support in Lean
   - automation approaches
2. Case study
   - Weak normalization of $\mathsf{STLC}_{\mathsf{num}}$
   - Strong normalization of $\mathsf{STLC}_{\mathsf{num}}$ $\vdash s : A \quad \rightarrow \mathrm{SN}\ s$

Strong Normalization — System

- $\mathsf{STLC}_{\mathsf{num}}$

$$
\begin{aligned}
A, B \in \mathsf{ty} &:= \mathsf{int} \mid A \to B \\
\Gamma \in \mathsf{ctx} &:= \emptyset \mid A \cdot \Gamma \\
s, t \in \mathsf{tm} &:= x \mid \lambda s \mid s\ t \mid \overline{n} \mid s + t \quad (x \in \mathbb{I}_m, n, m \in \mathbb{N})
\end{aligned}
$$

- Typing $\Gamma \vdash s : A$

$$\frac{\Gamma x = A}{\Gamma \vdash x : A} \qquad \frac{A \cdot \Gamma \vdash s : B}{\Gamma \vdash \lambda s : A \to B}$$

$$\frac{\Gamma \vdash s : A \to B \qquad \Gamma \vdash t : A}{\Gamma \vdash s\ t : B} \qquad \frac{\Gamma \vdash n_1 : \mathbb{N} \qquad \Gamma \vdash n_2 : \mathbb{N}}{\Gamma \vdash n_1 + n_2 : \mathbb{N}}$$

$$\frac{}{\Gamma \vdash \overline{n} : \mathbb{N}}$$

Strong Normalization — System

- Reduction $s \succ t$

$$\frac{s_1 \succ s_1'}{s_1\ s_2 \succ s_1'\ s_2} \qquad \frac{s_2 \succ s_2'}{s_1\ s_2 \succ s_1\ s_2'} \qquad \frac{s \succ s'}{\lambda s \succ \lambda s'}$$

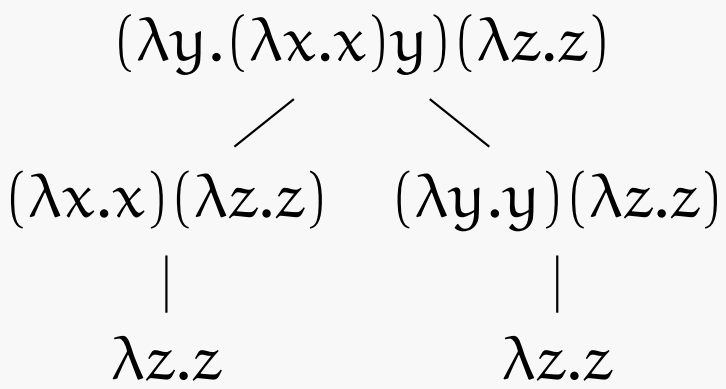
$$\frac{s_1 \succ s_1'}{s_1 + s_2 \succ s_1' + s_2} \qquad \frac{s_2 \succ s_2'}{s_1 + s_2 \succ s_1 + s_2'}$$

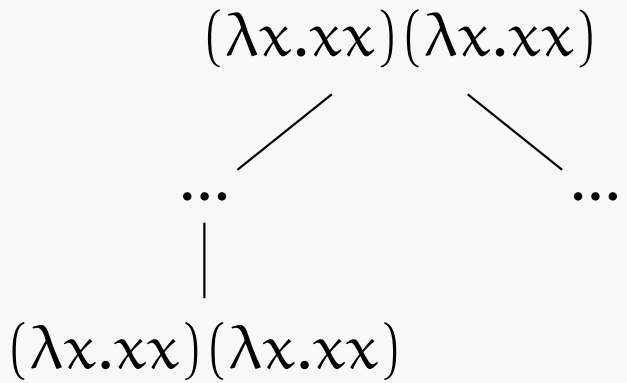
$$\frac{}{\lambda s\ t \succ s[t \cdot \mathrm{id}]} \qquad \frac{n_1 + n_2 = n_3}{\overline{n_1} + \overline{n_2} \succ \overline{n_3}}$$

- Reflexive-transitive closure of $\succ$

$$\frac{}{s \succ^* s} \qquad \frac{s_1 \succ s_2 \qquad s_2 \succ^* s_3}{s_1 \succ^* s_3}$$

Strong Normalization





- Accessibility predicate SN

$$\frac{\forall\, t.\ s \succ t \to \mathrm{SN}\ t}{\mathrm{SN}\ s}$$

Strong Normalization [Girard et al 1989]

$$\vdash s : A \xrightarrow{!} \mathrm{SN}$$

Strong Normalization [Girard et al 1989]

$$\vdash s : A \longrightarrow \models s : A \longrightarrow SN$$

$$\mathcal{R}$$

Strong Normalization [Girard et al 1989]

$$\vdash s : A \longrightarrow \models s : A \longrightarrow SN$$

$$\mathcal{R}$$

- Reducibility relation $\mathcal{R}$

$$\begin{aligned} \mathcal{R}_\Gamma[\![\mathbb{N}]\!] &:= \{s \mid \Gamma \vdash s : \mathbb{N} \ \wedge \ SN\ s\} \\ \mathcal{R}_\Gamma[\![A \to B]\!] &:= \{s \mid \Gamma \vdash s : A \to B \wedge \\ &\qquad \forall\ t.\ t \in \mathcal{R}_\Gamma[\![A]\!] \to s\ t \in \mathcal{R}_\Gamma[\![B]\!]\} \end{aligned}$$

Strong Normalization [Girard et al 1989]

$$\vdash s : A \longrightarrow \models s : A \longrightarrow SN$$

$$\mathcal{R}$$

- Reducibility relation $\mathcal{R}$

$$\begin{aligned}
\mathcal{R}_\Gamma[\![\mathbb{N}]\!] &:= \{s \mid \Gamma \vdash s : \mathbb{N} \ \wedge \ SN\ s\} \\
\mathcal{R}_\Gamma[\![A \to B]\!] &:= \{s \mid \Gamma \vdash s : A \to B \wedge \\
&\qquad \forall\ \xi\ \Delta\ t.\ \Gamma \preccurlyeq_\xi \Delta \to t \in \mathcal{R}_\Gamma[\![A]\!] \\
&\qquad \to (s\langle\xi\rangle\ t) \in \mathcal{R}_\Delta[\![B]\!]\}
\end{aligned}$$

Strong Normalization [Girard et al 1989]

$$\vdash s : A \longrightarrow \models s : A \xrightarrow{\mathcal{R}} SN$$

- Properties of $\mathcal{R}$
  - $\boldsymbol{CR_1}$. $s \in \mathcal{R}_\Gamma[\![A]\!] \to SN\ s$
  - $\boldsymbol{CR_2}$. $s \in \mathcal{R}_\Gamma[\![A]\!] \to s \succ^* t \to t \in \mathcal{R}_\Gamma[\![A]\!]$
  - $\boldsymbol{CR_3}$. $\Gamma \vdash s : A \to \text{neutral } s$
    $\to (\forall t.\ s \succ t \to t \in \mathcal{R}_\Gamma[\![A]\!]) \to s \in \mathcal{R}_\Gamma[\![A]\!]$

Strong Normalization [Girard et al 1989]

$$\vdash s : A \longrightarrow \models s : A \longrightarrow \mathrm{SN}$$

$$\mathcal{R}$$

- Properties of $\mathcal{R}$
- **Soundness** $\Gamma \vdash s : A \to s \in \mathcal{R}_\Gamma[\![A]\!]$

Strong Normalization [Girard et al 1989]

$$\vdash s : A \longrightarrow \models s : A \longrightarrow \mathrm{SN}$$

$$\mathcal{R}$$

- Properties of $\mathcal{R}$
- **Soundness** $\Gamma \vdash s : A \to \Gamma \preccurlyeq_\sigma \Delta \to s[\sigma] \in \mathcal{R}_\Delta[\![A]\!]$
- Strong normalization

Strong Normalization — Literature

- Girard-Tait proof technique [Girard et al 1989], Kripke-style logical relations [Mitchell et al 1988]
- LEGO [Altenkirch 1992]
- POPLMark Reloaded [Abel et al 2018]
- Coq, e.g. [Cooper 2015]:

> *"De Bruijn indices are foolishly difficult for this kind of proof"*

Strong Normalization [Girard et al 1989]

$$\vdash s : A \longrightarrow \models s : A \longrightarrow \mathrm{SN}$$

$$\mathcal{R}$$

- Properties of $\mathcal{R}$
- Soundness
- Strong normalization

} binders, substitution

Strong Normalization — CR_2

```
/- Reducibility properties: CR₂ -/
lemma CR₂' {n} (Γ) (A) (s t : tm n) :
  R Γ A s → s > t → R Γ A t :=
begin
    revert Γ s t n, induction A,
    { intros n Γ s t h q, split,
        { apply preservation; aauto_w ``(types_R), },
        constructor, intros t' ht', cases h.right,
        apply closed_star, apply star.sstep, exact ht',
        aauto_w ``(star.srefl), apply a, aauto },
    { intros n Γ s t h1 h2, split,
        { apply preservation, aauto, apply h1.left, },
        { intros n ξ Δ t p1 p2 ,
        have p3 := (h1.right ξ Δ t p1 p2),
        apply A_ih_a_1, assumption, constructor,
        aauto_w ``(substitutivity_ren),  }  }
end

theorem CR₂ {n} (Γ) (A) (s t : tm n) :
  R Γ A s → s >* t → R Γ A t :=
begin
  intros h q, induction q,
  { aauto, },
  { apply q_ih, apply (CR₂'); aauto }
end
```

Strong Normalization — CR_2

```
/- Reducibility properties: CR₂ -/
lemma CR₂' {n} (Γ) (A) (s t : tm n) :
  R Γ A s → s ≻ t → R Γ A t :=
begin
    revert Γ s t n, induction A,
    { intros n Γ s t h q, split,
        { apply preservation; aauto_w ``(types_R), },
        constructor, intros t' ht', cases h.right,
        apply closed_star, apply star.sstep, exact ht',
        aauto_w ``(star.srefl), apply a, aauto },
    { intros n Γ s t h1 h2, split,
        { apply preservation, aauto, apply h1.left, },
        { intros n ξ Δ t p1 p2 ,
        have p3 := (h1.right ξ Δ t p1 p2),
        apply A_ih_a_1, assumption, constructor,
        aauto_w ``(substitutivity_ren),  } }
end

theorem CR₂ {n} (Γ) (A) (s t : tm n) :
  R Γ A s → s ≻* t → R Γ A t :=
begin
  intros h q, induction q,
  { aauto, },
  { apply q_ih, apply (CR₂'); aauto }
end
```

Strong Normalization — substitution related

Lemma (Preservation)

$$\Gamma \vdash s : A \to s \succ^* t \to \Gamma \vdash s\ t : A$$

```
lemma preservation {n} (Γ) (s t : tm n) :
    s > t → forall A, Γ ⊢ s : A → Γ ⊢ t : A :=
begin
    intro hs, induction hs;
        intros A ht; cases ht;
        try { constructor}; aauto,

    { cases ht_a, apply morphism_subst,
      aauto, intro x, cases x, aauto,
      simp[scons], constructor }
end
```

Tactic State

```
n : ℕ,
s t : tm n,
hs_n : ℕ,
hs_e₁ : tm (succ hs_n),
hs_e₂ : tm hs_n,
Γ : Fin hs_n → type,
A ht_A : type,
ht_a_1 : Γ⊢hs_e₂:ht_A,
ht_a : Γ⊢lam hs_e₁:(ht_A↣A)
⊢ Γ⊢hs_e₁[hs_e₂..]:A
```

Strong Normalization – substitution related

Lemma (Preservation)

$$\Gamma \vdash s : A \to s \succ^* t \to \Gamma \vdash t : A$$

Lemma (Morphism for renamings)

$$\Gamma \vdash s : A \to \Gamma \preccurlyeq_\xi \Delta \to \Delta \vdash s\langle\xi\rangle : A$$

Definition (Agreement under renaming)

$$\Gamma \preccurlyeq_\xi \Delta := \forall x.\ \Delta(\xi\ x) = \Gamma\ x$$

Strong Normalization – substitution related

Lemma (Preservation)

$$\Gamma \vdash s : A \to s \succ^* t \to \Gamma \vdash t : A$$

Lemma (Morphism for renamings)

$$\Gamma \vdash s : A \to \Gamma \preccurlyeq_\xi \Delta \to \Delta \vdash s\langle\xi\rangle : A$$

Definition (Agreement under renaming)

$$\Gamma \preccurlyeq_\xi \Delta := \forall x.\ \Delta(\xi\ x) = \Gamma\ x$$

Strong Normalization — substitution related

```
/- Agreement of contexts. -/
definition agree_ren {n m} (Γ : ctx n) (Γ' : ctx m)
    (ξ : Fin m → Fin n)
    := forall x, (Γ (ξ x)) = Γ' x.

notation Γ `≼`:50 Δ `:`:50 ξ := agree_ren Δ Γ ξ.

/- Morphism for renaming. -/
lemma morphism_ren {m} (Γ : ctx m) (x : tm m) (A)
    (H : Γ ⊢ x : A ) :
∀ m' Δ (ξ : Fin m → Fin m'),
 (Γ ≼ Δ : ξ) → Δ ⊢ (ren_tm  ξ x) : A :=
begin
  induction H ; intros; aauto,
  {
    arw, specialize (a H_x),
    rw ← a, constructor },
  { arw, constructor; aauto },
  { arw, constructor; aauto },
  {
    arw,
    constructor,
    apply H_ih,
```

Updating

Strong Normalization — substitution related

```
/- Agreement of contexts. -/
definition agree_ren {n m} (Γ : ctx n) (Γ' : ctx m)
    (ξ : Fin m → Fin n)
    := forall x, (Γ (ξ x)) = Γ' x.

notation Γ `≼`:50 Δ `:`:50 ξ := agree_ren Δ Γ ξ.

/- Morphism for renaming. -/
lemma morphism_ren {m} (Γ : ctx m) (x : tm m) (A)
    (H : Γ ⊢ x : A ) :
∀ m' Δ (ξ : Fin m → Fin m'),
 (Γ ≼ Δ : ξ) → Δ ⊢ (ren_tm  ξ x) : A :=
begin
  induction H ; intros; aauto,
  {
    arw, specialize (a H_x),
    rw ← a, constructor },
  { arw, constructor; aauto },
  { arw, constructor; aauto },
  {
    arw,
    constructor,
    apply H_ih,
```

Tactic State — Updating

```
case types.tvar
m : ℕ,
Γ : ctx m,
x : tm m,
A : type,
H_m : ℕ,
H_Γ : Fin H_m → type,
H_x : Fin H_m,
m' : ℕ,
Δ : ctx m',
ξ : Fin H_m → Fin m',
a : H_Γ≼Δ:ξ
⊢ Δ⊢var_tm H_x.(ξ) :H_Γ H_x
```

Strong Normalization — substitution related

```
/- Agreement of contexts. -/
definition agree_ren {n m} (Γ : ctx n) (Γ' : ctx m)
    (ξ : Fin m → Fin n)
    := forall x, (Γ (ξ x)) = Γ' x.

notation Γ `≼`:50 Δ `:`:50 ξ := agree_ren Δ Γ ξ.

/- Morphism for renaming. -/
lemma morphism_ren {m} (Γ : ctx m) (x : tm m) (A)
    (H : Γ ⊢ x : A ) :
∀ m' Δ (ξ : Fin m → Fin m'),
 (Γ ≼ Δ : ξ) → Δ ⊢ (ren_tm  ξ x) : A :=
begin
  induction H ; intros; aauto,
  {
    arw, specialize (a H_x),
    rw ← a, constructor },
  { arw, constructor; aauto },
  { arw, constructor; aauto },
  {
    arw,
    constructor,
    apply H_ih,
```

Tactic State — Updating

```
m : ℕ,
Γ : ctx m,
x : tm m,
A : type,
H_m : ℕ,
H_Γ : Fin H_m → type,
H_x : Fin H_m,
m' : ℕ,
Δ : ctx m',
ξ : Fin H_m → Fin m',
a : H_Γ≼Δ:ξ
⊢ Δ⊢var_tm (ξ H_x):H_Γ H_x
```

step.lean:83:4: information trace output

```
✓ ren_tm.equations._eqn_1
```

Strong Normalization — substitution related

```
/- Agreement of contexts. -/
definition agree_ren {n m} (Γ : ctx n) (Γ' : ctx m)
    (ξ : Fin m → Fin n)
    := forall x, (Γ (ξ x)) = Γ' x.

notation Γ `≼`:50 Δ `:`:50 ξ := agree_ren Δ Γ ξ.

/- Morphism for renaming. -/
lemma morphism_ren {m} (Γ : ctx m) (x : tm m) (A)
    (H : Γ ⊢ x : A ) :
∀ m' Δ (ξ : Fin m → Fin m'),
 (Γ ≼ Δ : ξ) → Δ ⊢ (ren_tm  ξ x) : A :=
begin
  induction H ; intros; aauto,
  {
    simp with substLemmas, specialize (a H_x),
    rw ← a, constructor },
  { arw, constructor; aauto },
  { arw, constructor; aauto },
  {
    arw,
    constructor,
    apply H_ih,
```

Tactic State — Updating

```
case types.tvar
m : ℕ,
Γ : ctx m,
x : tm m,
A : type,
H_m : ℕ,
H_Γ : Fin H_m → type,
H_x : Fin H_m,
m' : ℕ,
Δ : ctx m',
ξ : Fin H_m → Fin m',
a : H_Γ≼Δ:ξ
⊢ Δ⊢var_tm (ξ H_x):H_Γ H_x
```

Strong Normalization — substitution related

```
/- Agreement of contexts. -/
definition agree_ren {n m} (Γ : ctx n) (Γ' : ctx m)
    (ξ : Fin m → Fin n)
    := forall x, (Γ (ξ x)) = Γ' x.

notation Γ `≼`:50 Δ `:`:50 ξ := agree_ren Δ Γ ξ.

/- Morphism for renaming. -/
lemma morphism_ren {m} (Γ : ctx m) (x : tm m) (A)
    (H : Γ ⊢ x : A ) :
∀ m' Δ (ξ : Fin m → Fin m'),
 (Γ ≼ Δ : ξ) → Δ ⊢ (ren_tm  ξ x) : A :=
begin
  induction H ; intros; aauto,
  {
    simp with substLemmas, specialize (a H_x),
    rw ← a, constructor },
  { arw, constructor; aauto },
  { arw, constructor; aauto },
  {
    arw,
    constructor,
    apply H_ih,
```

Tactic State — Updating

```
case types.tlam
m : ℕ,
Γ : ctx m,
x : tm m,
A : type,
H_m : ℕ,
H_Γ : Fin H_m → type,
H_e : tm (succ H_m),
H_A H_B : type,
H_a : (H_A.:H_Γ)⊢H_e:H_B,
H_ih : ∀ (m' : ℕ) (Δ : ctx m') (ξ : Fin (succ H_m) →
Fin m'), ((H_A.:H_Γ)≼Δ:ξ) → Δ⊢H_e.⟨ξ⟩ :H_B,
m' : ℕ,
Δ : ctx m',
ξ : Fin H_m → Fin m',
a : H_Γ≼Δ:ξ
⊢ Δ⊢lam H_e.⟨ξ⟩ :(H_A↦H_B)
```

Strong Normalization — substitution related

```
/- Agreement of contexts. -/
definition agree_ren {n m} (Γ : ctx n) (Γ' : ctx m)
    (ξ : Fin m → Fin n)
    := forall x, (Γ (ξ x)) = Γ' x.

notation Γ `≼`:50 Δ `:`:50 ξ := agree_ren Δ Γ ξ.

/- Morphism for renaming. -/
lemma morphism_ren {m} (Γ : ctx m) (x : tm m) (A)
    (H : Γ ⊢ x : A ) :
∀ m' Δ (ξ : Fin m → Fin m'),
 (Γ ≼ Δ : ξ) → Δ ⊢ (ren_tm  ξ x) : A :=
begin
  induction H ; intros; aauto,
  {
    simp with substLemmas, specialize (a H_x),
    rw ← a, constructor },
  { arw, constructor; aauto },
  { arw, constructor; aauto },
  {
    arw,
    constructor,
    apply H_ih,
```

Tactic State — Updating

```
m : ℕ,
Γ : ctx m,
x : tm m,
A : type,
H_m : ℕ,
H_Γ : Fin H_m → type,
H_e : tm (succ H_m),
H_A H_B : type,
H_a : (H_A.:H_Γ)⊢H_e:H_B,
H_ih : ∀ (m' : ℕ) (Δ : ctx m') (ξ : Fin (succ H_m) →
Fin m'), ((H_A.:H_Γ)≼Δ:ξ) → Δ⊢H_e.⟨ξ⟩ :H_B,
m' : ℕ,
Δ : ctx m',
ξ : Fin H_m → Fin m',
a : H_Γ≼Δ:ξ
⊢ Δ⊢lam H_e.⟨((Fin.fz.:ξ>>↑))⟩ :(H_A→H_B)
```

step.lean:88:4: information trace output

```
✓ ren_tm.equations._eqn_3
✓ upRen_tm_tm.equations._eqn_1
✓ up_ren.equations._eqn_1
✓ var_zero.equations._eqn_1
```

Strong Normalization — substitution related

Lemma (Preservation)

$$\Gamma \vdash s : A \to s \succ^* t \to \Gamma \vdash s\ t : A$$

```
lemma preservation {n} (Γ) (s t : tm n) :
    s > t → forall A, Γ ⊢ s : A → Γ ⊢ t : A :=
begin
    intro hs, induction hs;
        intros A ht; cases ht;
        try { constructor}; aauto,

    { cases ht_a, apply morphism_subst,
      aauto, intro x, cases x, aauto,
      simp[scons], constructor }
end
```

Tactic State

```
3 goals
n : ℕ,
s t : tm n,
hs_n : ℕ,
hs_e₁ : tm (succ hs_n),
hs_e₂ : tm hs_n,
Γ : Fin hs_n → type,
A ht_A : type,
ht_a_1 : Γ⊢hs_e₂:ht_A,
ht_a_a : (ht_A.:Γ)⊢hs_e₁:A
⊢ ?m_1⊢hs_e₁:A

n : ℕ,
s t : tm n,
hs_n : ℕ,
hs_e₁ : tm (succ hs_n),
hs_e₂ : tm hs_n,
Γ : Fin hs_n → type,
A ht_A : type,
ht_a_1 : Γ⊢hs_e₂:ht_A,
ht_a_a : (ht_A.:Γ)⊢hs_e₁:A
⊢ agree_subst_types ?m_1 Γ (hs_e₂.:var_tm)
```

Strong Normalization — Formalisations

	Lean		*Coq*	
Components	**Def.**	**Proof**	**Def.**	**Proof**
Autosubst	167	160	194	57
Reduction and Typing	60	135	51	119
Weak Normalization	10	97	17	73
Strong Normalization	19	350	28	392
Meta, `Ltac`	~ 160	–	~ 90	–
$\sum$	89	582	96	584

Future Work

- More complicated languages and proofs
  - POPLMark: System F
- Expressivity
  - Mutually inductive sorts
  - Lean 4?
- AS for other theorem provers
  - e.g. Agda
- Automation
  - Reification

Contributions

1. Adaptation of Autosubst to Lean
   - Tool for binder support in Lean
   - Automation approaches

2. Case study
   - Weak normalization of $\mathrm{STLC}_{\mathrm{num}}$
   - Strong normalization of $\mathrm{STLC}_{\mathrm{num}}$

`http://www.ps.uni-saarland.de/~mameche/bachelor.php`

AS in Lean — Tactics

```
--rewriting and unfolding (once)
meta def arw1 : tactic unit :=
  do
  (rw_unfoldables (rw_exprs tactic.failed) Eqns) <|>
  (rw_lemmas (rw_exprs tactic.failed) RwLemmas) <|>
  (rw_unfoldables (rw_exprs tactic.failed) EqnsUp) <|>
  (rw_unfoldables (rw_exprs tactic.failed) EqnsFin) <|>
  (rw_lemmas (rw_exprs tactic.failed) RwLemmasFin)

--without unfolding
meta def arw_cautious : tactic unit :=
  do
    (rw_lemmas (rw_exprs tactic.skip) RwLemmas) <|>
    (rw_lemmas (rw_exprs tactic.skip) RwLemmasFin)

--rewriting
meta def arw : tactic unit := tactic.repeat arw1
meta def arw' : tactic unit := tactic.repeat arw_cautious
```

AS in Lean — Tactics

```
meta def now_arw : tactic unit := do arw, tactic.reflexivity

--use hint
meta def arw_using : list pexpr → tactic unit
| e := do arw, rw_exprs tactic.skip tt e

--arw in hypothesis
meta def arw_at (h) : tactic unit :=
do
  hyp ← tactic.get_local h,
  tactic.revert hyp,
  arw,
  tactic.intro h,
  tactic.skip
```

Weak Normalization — Steps

- Simply typed λ-calculus, small and big step semantics
- Define
  - semantic interpretation of values, expressions, and contexts
  - semantic typing $\models$
- Prove
  - Soundness $\vdash \subseteq \models$
  - Weak normalization

$$\vdash s : A \to \exists\, v,\ s \succ^* v$$

Weak normalization — Logical Relations

$$
\begin{aligned}
\mathcal{V}[\![\mathbb{N}]\!] &:= \{\overline{n}\} \\
\mathcal{V}[\![A \to B]\!] &:= \{\lambda s \mid \forall v \in \mathcal{V}[\![A]\!].\ s[v \cdot \mathrm{id}] \in \mathcal{E}[\![B]\!]\} \\
\mathcal{E}[\![A]\!] &:= \{s \mid \exists t,\ s \succ^* t \wedge t \in \mathcal{V}[\![A]\!]\} \\
\mathcal{G}[\![\Gamma]\!] &:= \{\sigma \mid \forall x.\ \sigma x \in \mathcal{V}[\![\Gamma x]\!]\}
\end{aligned}
$$

Semantic typing

$$\Gamma \vDash s : A := \forall \sigma \in \mathcal{G}[\![\Gamma]\!].\ s[\sigma] \in \mathcal{E}[\![A]\!]$$