Simply typed β-convertibility is Tower-complete even for safe λ-terms

Nguyễn Lê Thành Dũng (a.k.a. Tito) — nltd@nguyentito.eu
École normale supérieure de Lyon

16th workshop on Computational Logic and Applications, January 13th, 2023

Dynamics of the untyped λ-calculus

$$t, u ::= \underbrace{x}_{\text{variables: } x, y, z\ldots} \mid t\ u \mid \lambda x.\ t \qquad\qquad (\lambda x.\ t)u \longrightarrow_\beta t\{x := u\}$$

$$(\lambda x.\ x\ x)\ (\lambda x.\ x\ x) \longrightarrow_\beta (x\ x)\{x := (\lambda x.\ x\ x)\} = (\lambda x.\ x\ x)\ (\lambda x.\ x\ x) \longrightarrow_\beta \ldots$$

Dynamics of the untyped λ-calculus

$$t, u ::= \underbrace{x}_{\text{variables: } x, y, z\ldots} \mid t\ u \mid \lambda x.\ t \qquad\qquad (\lambda x.\ t)u \longrightarrow_\beta t\{x := u\}$$

$$(\lambda x.\ x\ x)\ (\lambda x.\ x\ x) \longrightarrow_\beta (x\ x)\{x := (\lambda x.\ x\ x)\} = (\lambda x.\ x\ x)\ (\lambda x.\ x\ x) \longrightarrow_\beta \ldots$$

- Non-termination possible – actually, the untyped λ-calculus is Turing-complete
- *Convertibility* $t =_\beta u$ (reflexive transitive closure of $\to_\beta$) is undecidable

Dynamics of the untyped λ-calculus

$$t, u ::= \underbrace{x}_{\text{variables: } x,y,z\ldots} \mid t\ u \mid \lambda x.\ t \qquad\qquad (\lambda x.\ t)u \longrightarrow_\beta t\{x := u\}$$

$$(\lambda x.\ x\ x)\ (\lambda x.\ x\ x) \longrightarrow_\beta (x\ x)\{x := (\lambda x.\ x\ x)\} = (\lambda x.\ x\ x)\ (\lambda x.\ x\ x) \longrightarrow_\beta \ldots$$

- Non-termination possible – actually, the untyped λ-calculus is Turing-complete
- *Convertibility* $t =_\beta u$ (reflexive transitive closure of $\to_\beta$) is undecidable

This talk:

- Recall/clarify the complexity of $t =_\beta u$ for *simply typed* λ-terms
- Then extend the result to *safe* λ-terms

The simply typed λ-calculus

We now consider a *type system*: labeling λ-terms with specifications

$$t : A \to B \quad \approx \quad \text{“}t\text{ is a function from }A\text{ to }B\text{”}$$

Simple types: built using "$\to$" from a base type o

The simply typed λ-calculus

We now consider a *type system*: labeling λ-terms with specifications

$$t : A \to B \quad \approx \quad \text{“}t \text{ is a function from } A \text{ to } B\text{”}$$

Simple types: built using “$\to$” from a base type o

$$\dfrac{f : o \to o \qquad \dfrac{f : o \to o \qquad x : o}{f\,x : o}}{f\,(f\,x) : o}$$

The simply typed λ-calculus

We now consider a *type system*: labeling λ-terms with specifications

$$t : A \to B \quad \approx \quad \text{“}t\text{ is a function from }A\text{ to }B\text{”}$$

Simple types: built using “$\to$” from a base type o

$$\dfrac{f : o \to o \qquad \dfrac{f : o \to o \qquad x : o}{f\,x : o}}{f\,(f\,x) : o} \qquad \underbrace{\lambda f.\ \lambda x.\ f\,(f\,x)}_{\text{encodes the number 2}} : \overbrace{(o \to o) \to o \to o}^{\text{“type of natural numbers”}}$$

The simply typed λ-calculus

We now consider a *type system*: labeling λ-terms with specifications

$$t : A \to B \quad \approx \quad \text{“}t \text{ is a function from } A \text{ to } B\text{”}$$

Simple types: built using “$\to$” from a base type o

$$\dfrac{f : o \to o \qquad \dfrac{f : o \to o \qquad x : o}{f\,x : o}}{f\,(f\,x) : o} \qquad \underbrace{\lambda f.\,\lambda x.\,f\,(f\,x)}_{\text{encodes the number 2}} : \overbrace{(o \to o) \to o \to o}^{\text{“type of natural numbers”}}$$

Theorem

$\to_\beta$ is strongly normalizing (terminating) and confluent on simply typed λ-terms.

Corollary

$=_\beta$ is decidable on simply typed λ-terms.

Just compute the normal forms and compare them!

Two related questions

Theorem

$\to_\beta$ is strongly normalizing (terminating) and confluent on simply typed λ-terms.

Corollary

$=_\beta$ is decidable on simply typed λ-terms (compute & compare normal forms).

- Combinatorics: what's the length of $\to_\beta$ sequences?

  (for *linear* λ-terms: see Alexandros Singh's PhD thesis)
- Computational complexity: how hard is it to decide $=_\beta$?

What's the length of $\to_\beta$ sequences?

- Schwichtenberg, *Complexity of Normalization in the Pure Typed Lambda-Calculus*, 1982
- Beckmann, *Exact bounds for lengths of reductions in typed λ-calculus*, 2001

What's the length of $\to_\beta$ sequences?

- Schwichtenberg, *Complexity of Normalization in the Pure Typed Lambda-Calculus*, 1982
- Beckmann, *Exact bounds for lengths of reductions in typed λ-calculus*, 2001

Let the *order* of a type be the maximum nesting of $\to$ to the left:

$$\mathrm{ord}(o) = 0 \qquad \mathrm{ord}(A \to B) = \max(\mathrm{ord}(A) + 1, \mathrm{ord}(B))$$

Theorem

Let t be a simply typed λ-term. If all subterms of t have types of order $\leq k$, *then the maximum reduction length for t is at most $2_n(O(\mathrm{size}(t)))$ where $2_0(x) = x$ and $2_{k+1}(x) = 2^{2_k(x)}$.*

$\rightsquigarrow$ *tower of exponentials of height k*

What's the length of $\to_\beta$ sequences?

- Schwichtenberg, *Complexity of Normalization in the Pure Typed Lambda-Calculus*, 1982
- Beckmann, *Exact bounds for lengths of reductions in typed λ-calculus*, 2001

Let the *order* of a type be the maximum nesting of $\to$ to the left:

$$\mathrm{ord}(o) = 0 \qquad \mathrm{ord}(A \to B) = \max(\mathrm{ord}(A)+1, \mathrm{ord}(B))$$

Theorem

Let t be a simply typed λ-term. If all subterms of t have types of order $\leq k$, *then the maximum reduction length for t is at most* $2_n(O(\mathrm{size}(t)))$ *where* $2_0(x) = x$ *and* $2_{k+1}(x) = 2^{2_k(x)}$.

$\rightsquigarrow$ *tower of exponentials of height k*

$$\underbrace{(\lambda f.\, \lambda x.\, f\,(f\,x))\ \ldots\ \overbrace{(\lambda f.\, \lambda x.\, f\,(f\,x))}^{\text{type } (o\to o)\to(o\to o) \text{ of order } 2}}_{n \text{ times}}\,(\lambda x.\, g\ x\ x)\ y \quad \longrightarrow_\beta^* \quad \text{something of size } \Omega\left(2^{2^n}\right)$$

What's the length of $\to_\beta$ sequences?

- Schwichtenberg, *Complexity of Normalization in the Pure Typed Lambda-Calculus*, 1982
- Beckmann, *Exact bounds for lengths of reductions in typed λ-calculus*, 2001

Let the *order* of a type be the maximum nesting of $\to$ to the left:

$$\mathrm{ord}(o) = 0 \qquad \mathrm{ord}(A \to B) = \max(\mathrm{ord}(A) + 1, \mathrm{ord}(B))$$

Theorem

Let t be a simply typed λ-term. If all subterms of t have types of order $\leq k$, *then the maximum reduction length for t is at most $2_n(O(\mathrm{size}(t)))$ where $2_0(x) = x$ and $2_{k+1}(x) = 2^{2_k(x)}$.*

$\rightsquigarrow$ *tower of exponentials of height k*

$$\underbrace{(\lambda f.\, \lambda x.\, f\,(f\,x))\ \ldots\ \overbrace{(\lambda f.\, \lambda x.\, f\,(f\,x))}^{\text{type } (o\to o)\to(o\to o) \text{ of order 2}}}_{n \text{ times}}\ (\lambda x.\, g\ x\ x)\ y \quad \longrightarrow_\beta^* \quad \text{something of size } \Omega\left(2^{2^n}\right)$$

- Asada, Kobayashi, Sin'ya & Tsukada, (presented at CLA 2019!)
  Almost Every Simply Typed Lambda-Term Has a Long Beta-Reduction Sequence, 2017

Reduction length vs complexity of convertibility

$\mathrm{ord}(o) = 0$ and $\mathrm{ord}(A \to B) = \max(\mathrm{ord}(A) + 1, \mathrm{ord}(B))$

Theorem

Let t be a simply typed λ-term. If all subterms of t have types of order $\leq k$*, then the maximum reduction length for t is at most* $2_n(O(\mathrm{size}(t)))$ *where* $2_0(x) = x$ *and* $2_{k+1}(x) = 2^{2_k(x)}$.

Compute normal forms and compare $\rightsquigarrow$ check $=_\beta$ in time $2_k(\mathrm{poly}(n))$ i.e. k-ExpTime

Reduction length vs complexity of convertibility

$\mathrm{ord}(o) = 0$ and $\mathrm{ord}(A \to B) = \max(\mathrm{ord}(A) + 1, \mathrm{ord}(B))$

Theorem

Let t be a simply typed λ-term. If all subterms of t have types of order $\leq k$, *then the maximum reduction length for t is at most $2_n(O(\mathrm{size}(t)))$ where $2_0(x) = x$ and $2_{k+1}(x) = 2^{2_k(x)}$.*

Compute normal forms and compare $\rightsquigarrow$ check $=_\beta$ in time $2_k(\mathrm{poly}(n))$ i.e. k-ExpTime

But one can do better! Let $\texttt{Bool} = o \to o \to o$, $\texttt{true} = \lambda x.\ \lambda y.\ x$ and $\texttt{false} = \lambda x.\ \lambda y.\ y$

Theorem (Terui 2012)

Fix $k \in \mathbb{N}$. Given a simply typed λ-term $t : \texttt{Bool}$ whose subterms have types of order $\leq 2k + 2$ (resp. $\leq 2k + 3$), checking whether $t =_\beta \texttt{true}$ is complete for k-ExpTime (resp. k-ExpSpace).

Reduction length vs complexity of convertibility

$\mathrm{ord}(o) = 0$ and $\mathrm{ord}(A \to B) = \max(\mathrm{ord}(A) + 1, \mathrm{ord}(B))$

Theorem

Let t be a simply typed λ-term. If all subterms of t have types of order $\leq k$, *then the maximum reduction length for t is at most* $2_n(O(\mathrm{size}(t)))$ *where* $2_0(x) = x$ *and* $2_{k+1}(x) = 2^{2_k(x)}$.

Compute normal forms and compare $\rightsquigarrow$ check $=_\beta$ in time $2_k(\mathrm{poly}(n))$ i.e. k-EXPTIME

But one can do better! Let `Bool` $= o \to o \to o$, `true` $= \lambda x.\ \lambda y.\ x$ and `false` $= \lambda x.\ \lambda y.\ y$

Theorem (Terui 2012)

Fix $k \in \mathbb{N}$. Given a simply typed λ-term t : `Bool` *whose subterms have types of order $\leq 2k + 2$ (resp. $\leq 2k + 3$), checking whether $t =_\beta$* `true` *is complete for k-EXPTIME (resp. k-EXPSPACE).*

Corollary (of the hardness part; originally from Statman 198X)

β-convertibility of arbitrary simply typed λ-terms is non-elementary, *i.e.* $\notin \bigcup_{k \in \mathbf{N}} k\text{-EXPTIME}$.

How hard is it to decide β-convertibility?

Theorem (Statman 198X)

β-convertibility of arbitrary simply typed λ-terms is non-elementary, *i.e.* $\notin \bigcup_{k \in \mathbf{N}} k\text{-}\textsc{ExpTime}$.

At the same time, we can check it naively in $2_{\mathrm{order}}(\mathrm{poly}(\mathrm{size}))$:

tower of exponentials of "reasonably" increasing height, *"just beyond" elementary*

So can we be more precise?

How hard is it to decide β-convertibility?

Theorem (Statman 198X)

β-convertibility of arbitrary simply typed λ-terms is non-elementary, *i.e.* $\notin \bigcup_{k \in \mathbf{N}} k\text{-}\textsc{ExpTime}$.

At the same time, we can check it naively in $2_{\text{order}}(\text{poly}(\text{size}))$:

tower of exponentials of "reasonably" increasing height, *"just beyond" elementary*

So can we be more precise?

Definition (Schmitz 2016 – motivated by several problems in logic and verification)

$\textsc{Tower} = \bigcup_{f \text{ elementary}} \textsc{DTime}(2_{f(n)}(1))$ i.e. tower of exp of elementary height

Tower-completeness is defined w.r.t. elementary reductions.

Theorem

β-convertibility of arbitrary simply typed λ-terms is Tower-complete.

Simply typed β-convertibility is Tower-complete

Definition (Schmitz 2016)

$\textsc{Tower} = \bigcup_{f \text{ elementary}} \textsc{DTime}(2_{f(n)}(1))$ i.e. tower of exponentials of elementary height

Theorem (folklore, refining Statman 198X)

β-convertibility of arbitrary simply typed λ-terms is Tower*-complete.*

- Membership in Tower: naive algorithm (compare normal forms) is OK
- Tower-hard: reduction from "higher-order quantified boolean formulas" (Statman)

  e.g. $\forall f : \texttt{Bool} \to \texttt{Bool}.\ (\exists x : \texttt{Bool}.\ f(x)) \Rightarrow (\exists y : \texttt{Bool}.\ f(\texttt{not}(y)))$ is true

Simply typed β-convertibility is Tower-complete

Definition (Schmitz 2016)

$\textsc{Tower} = \bigcup_{f \text{ elementary}} \textsc{DTime}(2_{f(n)}(1))$ i.e. tower of exponentials of elementary height

Theorem (folklore, refining Statman 198X)

β-convertibility of arbitrary simply typed λ-terms is Tower-complete.

- Membership in Tower: naive algorithm (compare normal forms) is OK
- Tower-hard: reduction from "higher-order quantified boolean formulas" (Statman)

  e.g. $\forall f : \texttt{Bool} \to \texttt{Bool}.\ (\exists x : \texttt{Bool}.\ f(x)) \Rightarrow (\exists y : \texttt{Bool}.\ f(\texttt{not}(y)))$ is true
  - First supposed to appear in a never-existing paper [Fisher & Meyer 1975]
  - [Mairson 1992] gave a proof that it's non-elementary by simulating a Turing machine
  - Explicit Tower-hardness: Chistikov, Haase, Hadizadeh & Mansutti,

    Higher-Order Quantified Boolean Satisfiability, 2022

Simply typed β-convertibility is Tower-complete

Definition (Schmitz 2016)

$\textsc{Tower} = \bigcup_{f \text{ elementary}} \textsc{DTime}(2_{f(n)}(1))$ i.e. tower of exponentials of elementary height

Theorem (folklore, refining Statman 198X)

β-convertibility of arbitrary simply typed λ-terms is Tower-complete.

- Membership in Tower: naive algorithm (compare normal forms) is OK
- Tower-hard: reduction from "higher-order quantified boolean formulas" (Statman)

  e.g. $\forall f : \texttt{Bool} \to \texttt{Bool}.\ (\exists x : \texttt{Bool}.\ f(x)) \Rightarrow (\exists y : \texttt{Bool}.\ f(\texttt{not}(y)))$ is true
  - First supposed to appear in a never-existing paper [Fisher & Meyer 1975]
  - [Mairson 1992] gave a proof that it's non-elementary by simulating a Turing machine
  - Explicit Tower-hardness: Chistikov, Haase, Hadizadeh & Mansutti, *Higher-Order Quantified Boolean Satisfiability*, 2022

`https://cstheory.stackexchange.com/questions/34883/reference-request-deciding-validity-of-higher-order-quantified-boolean-formulas`

Motivating the safety condition

The *safety* restriction on simply typed λ-terms comes from the theory of *higher-order recursion schemes* which generate infinite trees

(also the motivation of Asada et al.'s work on average-case reduction length)

simply typed λ-terms + `let rec` $\equiv$ collapsible pushdown automata (late 2000s tech)

safe λ-terms + `let rec` $\equiv$ higher-order pushdown automata (from 1980s)

[Damm '82; Knapkik, Niwiński & Urzyczyn '02; Salvati & Walukiewicz '12]

Example on next slide, but it's not important for the rest of the talk, just for motivation

Generating infinite trees: automata vs λ-calculus

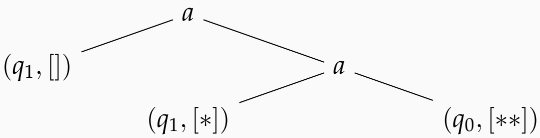
$(q_0, [])$

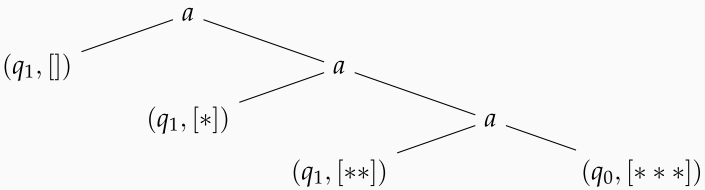
Generating infinite trees: automata vs λ-calculus

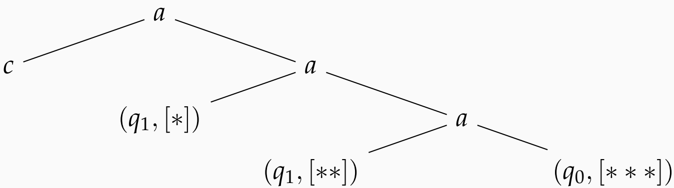
$$a$$

$$(q_1, [\,]) \qquad\qquad (q_0, [*])$$

Generating infinite trees: automata vs λ-calculus

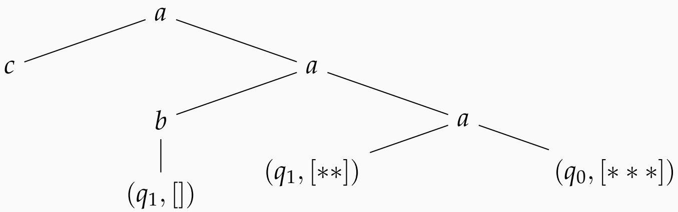



a
$(q_1, [])$
a
$(q_1, [*])$
a
$(q_1, [**])$
$(q_0, [***])$

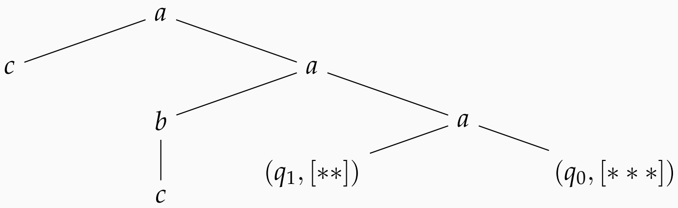


a
c
a
$(q_1, [*])$
a
$(q_1, [**])$
$(q_0, [***])$

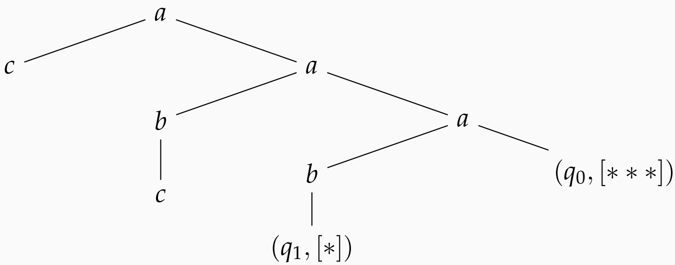
Generating infinite trees: automata vs λ-calculus

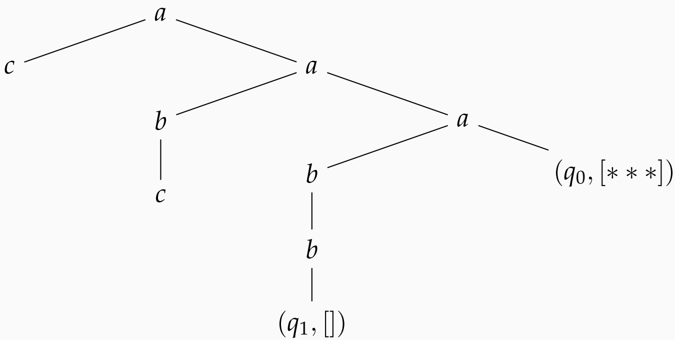


Generating infinite trees: automata vs λ-calculus

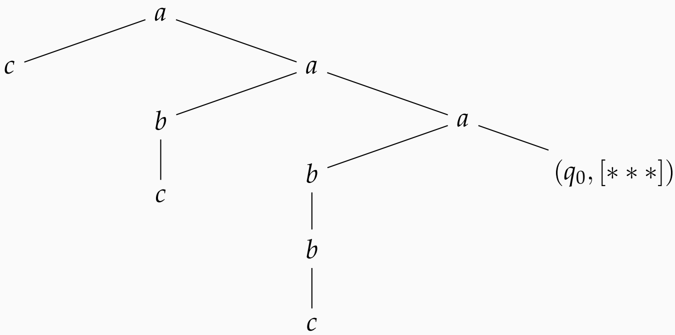


Generating infinite trees: automata vs λ-calculus

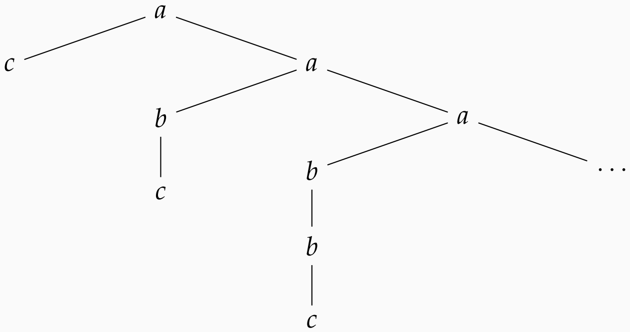



a
c
a
b
a
c
b
$(q_0, [***])$
b
$(q_1, [])$

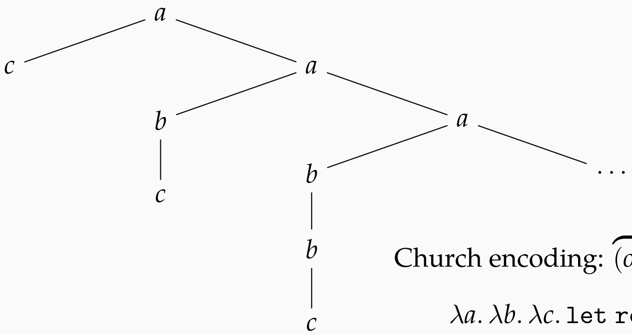
Generating infinite trees: automata vs λ-calculus



Generating infinite trees: automata vs λ-calculus



Generating infinite trees: automata vs λ-calculus



Church encoding: $\overbrace{(o \to o \to o)}^{a} \to \overbrace{(o \to o)}^{b} \to \overset{\overset{c}{\downarrow}}{o} \to o$

$$\lambda a.\, \lambda b.\, \lambda c.\, \texttt{let rec}\, f = \lambda x.\, a\, x\, (f\, (b\, x))\ \texttt{in}\, f\, c$$

The safe λ-calculus (without `let rec`)

Definition (Blum & Ong 2009)

A simply typed λ-term t is *unsafe* if it contains a subterm t' such that

- t' contains some x as a free variable with $\mathrm{ord}(x) < \mathrm{ord}(t')$
- t' is not applied to another subterm: $t = C[\lambda x.\ t']$ or $t = C[u\ t']$

$\lambda f^{(o \to o) \to o}.\ f\ (\lambda x^o.\ f\ (\lambda y^o.\ y))$ is safe

$$\lambda f^{(o \to o) \to o}.\ f\ (\lambda x^o.\ f\ \underbrace{(\lambda y^o.\ \overset{\text{type } o \ \rightsquigarrow\ \text{order } 0}{\overset{\downarrow}{x}})}_{\text{type } (o \to o)\ \rightsquigarrow\ \text{order } 1})$$ is unsafe!

The safe λ-calculus (without `let rec`)

Definition (Blum & Ong 2009)

A simply typed λ-term t is *unsafe* if it contains a subterm t' such that

- t' contains some x as a free variable with $\mathrm{ord}(x) < \mathrm{ord}(t')$
- t' is not applied to another subterm: $t = C[\lambda x.\ t']$ or $t = C[u\ t']$

$\lambda f^{(o\to o)\to o}.\ f\ (\lambda x^o.\ f\ (\lambda y^o.\ y))$ is safe

$\lambda f^{(o\to o)\to o}.\ f\ (\lambda x^o.\ f\ (\underbrace{\lambda y^o.\ \overset{\text{type } o\ \rightsquigarrow\ \text{order } 0}{x}}_{\text{type } (o\to o)\ \rightsquigarrow\ \text{order } 1}))$ is unsafe!

Theorem (Blum & Ong 2009)

β-convertibility is PSPACE-hard on safe λ-terms.

By reduction from Quantified Boolean Formulas…

but they don't manage to encode *higher-order* QBF without violating safety!

Complexity of β-convertibility in the safe fragment

Theorem (Blum & Ong 2009)

β-convertibility is PSPACE-hard on safe λ-terms.

They also remark:

> *Because the safety condition restricts expressivity in a non-trivial way, one can reasonably expect the beta-eta equivalence problem to have a lower complexity in the safe case than in the normal case; this intuition is strengthened by our failed attempt to encode [higher-order QBF] in the safe lambda calculus. No upper bounds is known at present. On the other hand our PSPACE-hardness result is probably a coarse lower bound; it would be interesting to know whether we also have EXPTIME-hardness.*

Complexity of β-convertibility in the safe fragment

Theorem (Blum & Ong 2009)

β-convertibility is PSPACE-hard on safe λ-terms.

They also remark:

> *Because the safety condition restricts expressivity in a non-trivial way, one can reasonably expect the beta-eta equivalence problem to have a lower complexity in the safe case than in the normal case; this intuition is strengthened by our failed attempt to encode [higher-order QBF] in the safe lambda calculus. No upper bounds is known at present. On the other hand our PSPACE-hardness result is probably a coarse lower bound; it would be interesting to know whether we also have EXPTIME-hardness.*

Theorem (new!)

β-convertibility is Tower-complete on safe λ-terms.

So it's not any easier in the safe case than in the normal case! (at this level of precision)

Our new Tower-hardness proof

Theorem (new!)

β-convertibility is Tower*-complete on safe λ-terms.*

Blum and Ong didn't manage to reduce from higher-order QBF

$\rightsquigarrow$ we take another approach: a reduction from *star-free expression* emptiness

$$E, E' ::= \varnothing \mid \overbrace{\varepsilon}^{\text{empty string}} \mid \underbrace{a}_{\text{letter in a finite alphabet } \Sigma} \mid E \cup E' \mid \overbrace{E \cdot E'}^{\text{concatenation}} \mid \underbrace{\neg E}_{\text{complement}} \qquad \rightsquigarrow \qquad [\![E]\!] \subseteq \Sigma^*$$

(Remark: *star-free languages* = a subclass of regular languages also characterized by aperiodic monoids, first-order logic over finite models, linear temporal logic, ...)

Our new Tower-hardness proof

Theorem (new!)

β-convertibility is Tower-complete on safe λ-terms.

Blum and Ong didn't manage to reduce from higher-order QBF

$\rightsquigarrow$ we take another approach: a reduction from *star-free expression* emptiness

$$E, E' ::= \varnothing \mid \overbrace{\varepsilon}^{\text{empty string}} \mid \underbrace{a}_{\text{letter in a finite alphabet } \Sigma} \mid E \cup E' \mid \overbrace{E \cdot E'}^{\text{concatenation}} \mid \underbrace{\neg E}_{\text{complement}} \qquad \rightsquigarrow \qquad [\![E]\!] \subseteq \Sigma^*$$

(Remark: *star-free languages* = a subclass of regular languages also characterized by aperiodic monoids, first-order logic over finite models, linear temporal logic, ...)

Theorem (Stockmeyer & Meyer 1973, revisited by Schmitz 2016)

Given a star-free expression E, it is Tower-complete to decide whether $[\![E]\!] = \varnothing$.

source of complexity: alternation $\cdot$ vs $\neg$ (cf. dot-depth / Straubing–Thérien hierarchy)

Turning star-free expressions into safe λ-terms

$$E, E' ::= \varnothing \mid \varepsilon \mid a \mid E \cup E' \mid E \cdot E' \mid \neg E \qquad \rightsquigarrow \qquad [\![E]\!] \subseteq \Sigma^*$$

Lemma

Any expression E can be turned in PTIME into an equivalent safe term $t_E : \mathtt{Str}_\Sigma[A] \to \mathtt{Bool}$.

$$\mathtt{Str}_\Sigma[A] = \overbrace{(A \to A) \to \cdots \to (A \to A)}^{|\Sigma| \text{ times}} \to A \to A$$

$$abb \in \Sigma^* = \{a, b\}^* \quad \rightsquigarrow \quad \overline{abb} = \lambda f_a.\ \lambda f_b.\ \lambda x.\ f_a\ (f_b\ (f_b\ x)) : \mathtt{Str}_\Sigma[A] \quad \text{for any } A$$

Turning star-free expressions into safe λ-terms

$$E, E' ::= \varnothing \mid \varepsilon \mid a \mid E \cup E' \mid E \cdot E' \mid \neg E \qquad \rightsquigarrow \qquad [\![E]\!] \subseteq \Sigma^*$$

Lemma

Any expression E can be turned in PTIME into an equivalent safe term $t_E : \mathtt{Str}_\Sigma[A] \to \mathtt{Bool}$.

$$\mathtt{Str}_\Sigma[A] = \overbrace{(A \to A) \to \cdots \to (A \to A)}^{|\Sigma| \text{ times}} \to A \to A$$

$$abb \in \Sigma^* = \{a, b\}^* \quad \rightsquigarrow \quad \overline{abb} = \lambda f_a.\, \lambda f_b.\, \lambda x.\, f_a\ (f_b\ (f_b\ x)) : \mathtt{Str}_\Sigma[A] \quad \text{for any } A$$

Theorem (Hillebrand & Kanellakis 1996 – main inspiration for the above)

A language $L \subseteq \Sigma^*$ can be defined by some simply typed λ-term $t : \mathtt{Str}_\Sigma[A] \to \mathtt{Bool}$ (where A may be chosen depending on L) if and only if it is regular.

Example: $t = \lambda s.\, s\ \mathtt{id}\ \mathtt{not}\ \mathtt{true} : \mathtt{Str}_{\{a,b\}}[\mathtt{Bool}] \to \mathtt{Bool}$ (even number of bs)

$$t\ \overline{abb} \longrightarrow_\beta \overline{abb}\ \mathtt{id}\ \mathtt{not}\ \mathtt{true} \longrightarrow_\beta \mathtt{id}\ (\mathtt{not}\ (\mathtt{not}\ \mathtt{true})) \longrightarrow_\beta \mathtt{true}$$

Turning star-free expressions into safe λ-terms

Lemma

Any expression E can be turned in PTIME into an equivalent safe term $t_E : \mathtt{Str}_\Sigma[A] \to \mathtt{Bool}$.

The proof is inspired by my research with Pradic on "Implicit automata in typed λ-calculi"

- Especially our results on *transducers* (automata computing string-to-string functions)
- e.g. the inductive case for translating $E \cdot E'$ uses a safe λ-term computing

$$\mathtt{123} \mapsto \square\mathtt{123}\#\mathtt{1}\square\mathtt{23}\#\mathtt{12}\square\mathtt{3}\#\mathtt{123}\square$$

  a typical *polyregular function* (cf. Bojańczyk's LICS'22 invited paper)

Then we need a bit more work to apply t_E to all "short enough" words and take the disjunction. Finally we get a term of type $\mathtt{Bool}$ which is $\mathtt{true}$ when $[\![E]\!] \neq \varnothing$, hence:

Theorem

Given a safe λ-term $t : \mathtt{Bool}$, it is TOWER-complete to decide whether $t =_\beta \mathtt{true}$.

Conclusion

Theorem

Given a safe λ-term t : `Bool`, it is Tower*-complete to decide whether* $t =_\beta$ `true`.

- The same holds for the simply typed λ-calculus (of which the safe λ-calculus is a fragment), "traditionally" proved via higher-order quantified boolean formulas
- This does not work in the safe case; instead we leverage connections between automata theory and λ-calculus

Conclusion

Theorem

Given a safe λ-term $t : \texttt{Bool}$, it is Tower*-complete to decide whether $t =_\beta \texttt{true}$.*

- The same holds for the simply typed λ-calculus (of which the safe λ-calculus is a fragment), "traditionally" proved via higher-order quantified boolean formulas
- This does not work in the safe case; instead we leverage connections between automata theory and λ-calculus

Final remark: Pradic and I have characterized star-free languages using *planar* λ-terms (ICALP 2020). For this result, translating star-free expressions didn't work (instead we used the Krohn–Rhodes decomposition theorem).