Program termination in the simply-typed $\lambda$-calculus

Frédéric Blanqui

INRIA

2nd Asian-Pacific Summer School on Formal Methods
Tsinghua University, 20-27 August 2010
Outline

\(\beta\)-reduction

Van Daalen’s proof (1980)

Tait’s proof (1967)
Untyped $\lambda$-calculus

introduced by Alonzo Church in 1932

$\lambda$-terms: $t \in L = x \in X \mid \lambda x.t \mid tt$
\( \rightarrow_\beta \) is defined by induction as follows:

\[
\begin{align*}
\text{(top)} & \quad (\lambda x.t)u \rightarrow_\beta t^u_x \\
\text{(abs)} & \quad \lambda x.t \rightarrow_\beta \lambda x.t' \\
\text{(app1)} & \quad t \rightarrow_\beta t' \\
& \quad tu \rightarrow_\beta t'u \\
\text{(app2)} & \quad u \rightarrow_\beta u' \\
& \quad tu \rightarrow_\beta tu'
\end{align*}
\]
Termination

A term $t$ terminates if every sequence of $\beta$-reductions starting from $t$ is finite.

i.e. there is no infinite sequence of $\beta$-reductions starting from $t$

$$t = t_0 \rightarrow_\beta t_1 \rightarrow_\beta t_2 \rightarrow_\beta \ldots$$

let $SN$ be the set of terminating terms
Which $\lambda$-terms terminate?

- $(\lambda x. \text{if } x \geq 2 \text{ then } t \text{ else } u)v$ is typable and terminates
- $(\lambda x. xx)(\lambda x. xx)$ is not typable and does not terminate
- $(\lambda x. \text{if } x \geq 2 \text{ then } t \text{ else } u)"foo"$ is not typable and terminates

Do all simply-typed $\lambda$-terms terminate?
Simply-typed \( \lambda \)-calculus à la Church simplified

**simple types:** \( S \in \mathcal{T} = B \in \mathcal{B} \mid S \rightarrow S \)

To remove the need for typing environments, we assume that each variable \( x \) is given a fixed type \( \tau_x \). Let \( \tau_t \) be the unique type of \( t \):

\[
\begin{align*}
(\text{var}) & \quad \frac{}{x : \tau_x} \\
(\text{abs}) & \quad \frac{t : T}{\lambda x.t : \tau_x \rightarrow T} \\
(\text{app}) & \quad \frac{u : S \rightarrow T \quad s : S}{us : T}
\end{align*}
\]

**Consequence:** induction on \( \nu : T \) is equivalent to induction on \( \nu \)
Definition: a substitution $\rho$ is well-typed if, for all $x$, $x\rho : \tau_x$.

Lemma: if $v : V$ and $\rho$ is well-typed, then $v\rho : V$.


Remark: In the following, we only consider well-typed terms and substitutions.
Preservation of typing under reduction

Lemma: if $v : V$ and $v \rightarrow_\beta v'$, then $v' : V$.

Outline

$\beta$-reduction

Van Daalen’s proof (1980)

Tait’s proof (1967)
First proof attempt

**Theorem:** if $\nu : V$, then $\nu \in SN$.

**Proof.** By induction on $\nu$.

- $\nu \in X$. Then, $\nu \in SN$.
- $\nu = \lambda x.t$. By IH, $t \in SN$. Thus, $\nu \in SN$. 

Frédéric Blanqui  Program termination in the simply-typed $\lambda$-calculus
Application case

- $v = us$. By IH, $u, s \in SN$. How to prove that $v = us \in SN$?

Remark: $v \in SN$ if every reduct of $v$ is $SN$.

Possible reducts of $us$:
- $u's$ with $u'$ a reduct of $u$
- $us'$ with $s'$ a reduct of $s$
- $t^s_x$ if $u = \lambda x.t$

Is every possible reduct $SN$?

Since $u, s \in SN$, the first two cases can be dealt with by well-founded induction on $(u, s)$.

For the third case, we need to strengthen the IH.
Second proof attempt

Theorem: if $v : V$ and $a : \tau_y$ are SN, then $v^a_y \in SN$.

Proof. Let $\rho = (a)_y$. By induction on $v$.

- $v \in \mathcal{X}$. If $v = y$, then $v\rho = a \in SN$. Otherwise, $v\rho = v \in SN$.

- $v = \lambda x.t$. Wlog we can assume that $x \neq y$ and $x \notin FV(a)$. Thus, $v\rho = \lambda x.t\rho$. By IH, $t\rho \in SN$. Thus, $v\rho \in SN$.
Application case: $v = u ss_1 \ldots s_n$

- $u = x \neq y$. Then, $v \rho = xs \rho s_1 \rho \ldots s_n \rho \in SN$ since, by IH, $s \rho, s_1 \rho, \ldots, s_n \rho \in SN$.

- $u = y$. Then, $v \rho = as \rho s_1 \rho \ldots s_n \rho$. If $v \rho \notin SN$, then $a \rightarrow^* \lambda x. b$ and $b^x s_1 \rho \ldots s_n \rho \notin SN$. How to conclude?

  Remark 1: $\tau_y = \tau_x \rightarrow \tau_z$
  Remark 2: for $c = b^x s_1 \rho$, we have $\tau_x < \tau_y$
  Remark 3: $b^x s_1 \rho \ldots s_n \rho = (zs_1 \rho \ldots s_n \rho)^c_z$ and $\tau_z < \tau_y$

- $u = \lambda x. t$. $v \rho = (\lambda x. t \rho)s \rho s_1 \rho \ldots s_n \rho$. We prove that $v \rho \in SN$ by well-founded induction on $(t, s, s_1, \ldots, s_n)$. Reducts of $v \rho$:
  
  - Reduction in $t \rho, s \rho, s_1 \rho, \ldots, s_n \rho$: IH.
  - Otherwise, the reduct is $t^{s \rho} s_1 \rho \ldots s_n \rho$. How to conclude?

    Remark 4: $t^{s \rho} s_1 \rho \ldots s_n \rho = (t^x s_1 \ldots s_n) \rho \leftarrow \beta v \rho$ and $v \in SN$
Final proof (Diederik Van Daalen, 1980)

**Theorem:** if $v : V$ and $a : \tau_y$ are $\text{SN}$, then $v^a_y \in \text{SN}$.

**Proof.** Let $\rho = (^a_y)$. By induction on $((\tau_y, v)$ using $\rightarrow_\beta \cup \triangleright$ as well-founded ordering on $v$.

- $v \in \mathcal{X}$. If $v = y$, then $v\rho = a \in \text{SN}$. Otherwise, $v\rho = v \in \text{SN}$.
- $v = \lambda x.t$. By IH, $t\rho \in \text{SN}$. Thus, $v\rho \in \text{SN}$.
- $v = us\vec{s}$. By IH, $s\rho, \vec{s}\rho \in \text{SN}$.
  - $u = x \neq y$. Then, $v\rho = xs\rho\vec{s}\rho \in \text{SN}$ since, by IH, $s\rho, \vec{s}\rho \in \text{SN}$.
  - $u = y$. Then, $v\rho = as\rho\vec{s}\rho$. If $v\rho \notin \text{SN}$, then $a \rightarrow^*_\beta \lambda x.b$ and $b^{s\rho}_{\vec{s}\rho} = (z\vec{s}\rho)^{b^s}_z \notin \text{SN}$. Since $b \in \text{SN}$ and $\tau_x < \tau_y$, by IH, $b^{s\rho}_{\vec{s}\rho} \in \text{SN}$. Since $z\vec{s}\rho \in \text{SN}$ and $\tau_z < \tau_y$, by IH, $b^{s\rho}_{\vec{s}\rho} \in \text{SN}$.
  - $u = \lambda x.t$. $v\rho = (\lambda x.t\rho)s\rho\vec{s}\rho$. Reducts of $v\rho$:
    - Reduction in $t\rho, s\rho, \vec{s}\rho$: IH.
    - Otherwise, the reduct is $t\rho^{s\rho}_{\vec{s}\rho} = (t^s_x\vec{s})\rho$. We have $t^s_x\vec{s} \in \text{SN}$ since it is a reduct of $v \in \text{SN}$. Thus, by IH, $t\rho^{s\rho}_{\vec{s}\rho} \in \text{SN}$. 

Frédéric Blanqui
Program termination in the simply-typed $\lambda$-calculus
Direct proof (Diederik Van Daalen, 1980)

- nice proof: created redexes have abstractions of decreasing types
- but we do not know how to extend it to richer type theories yet

From left to right: husband of Henriëtte, Jan van Hoek, Diederik van Daalen, Bert Jutting, Ids Zandleven, Roel de Vrijer, prof de Bruijn.
Outline

$\beta$-reduction

Van Daalen’s proof (1980)

Tait’s proof (1967)
Idea: strengthen the induction hypothesis again

Find a property $P$ on well-typed terms such that:

- if $P(v)$, then $v \in SN$
- if $P(u : S \to T)$ and $P(s : S)$, then $P(us)$
- if $P(u)$ and $P(s)$, then $P(u^s_x)$
- $P(x)$ holds for every variable $x$
William Walker Tait’s approach (1967)

\( u : V \) is computable if:

- either \( V \in B \) and \( u \in SN \)
- or \( V = S \rightarrow T \) and, for all computable \( s : S \), \( us \) is computable

This provides an inductive interpretation of types:

- \( \llbracket B \rrbracket = \{ u : B \mid u \in SN \} \) is the set of computable terms of type \( B \)
- \( \llbracket S \rightarrow T \rrbracket = \{ u : S \rightarrow T \mid \forall s \in \llbracket S \rrbracket, us \in \llbracket T \rrbracket \} \) is the set of computable terms of type \( S \rightarrow T \)

A substitution \( \rho \) is computable if, for all \( x \), \( x\rho \in \llbracket \tau_x \rrbracket \)
Computability, variables and termination

Let $X$ be the set of terminating terms of the form $xs_1 \ldots s_n$ ($n \geq 0$)

Lemma: For all type $V$, $X \subseteq (1) \left[ V \right] \subseteq (2) \text{SN}$.

Proof. By induction on $V$.

- $V \in B$.
  
  (1) Let $v \in X$. Since $X \subseteq \text{SN}$, $v \in \text{SN}$. Thus, $v \in \left[ V \right]$.
  
  (2) Let $v \in \left[ V \right]$. Then, $v \in \text{SN}$.

- $V = S \rightarrow T$.
  
  (1) Let $v = xs_1 \ldots s_n \in X$ and $s_{n+1} \in \left[ S \right]$. By IH2, $s_{n+1} \in \text{SN}$. Thus, $xs_1 \ldots s_{n+1} \in \text{SN}$. By IH2, $xs_1 \ldots s_{n+1} \in \left[ T \right]$. Thus, $v \in \left[ V \right]$.
  
  (2) Let $v \in \left[ V \right]$. By IH1, there is $x \in \left[ S \right]$. Thus, $vx \in \left[ T \right]$. By IH2, $vx \in \text{SN}$. Thus, $v \in \text{SN}$.
Tait’s approach

Lemma: If $\nu : V$ and $\rho$ is computable, then $\nu \rho \in \llbracket V \rrbracket$.

Proof. By induction on $\nu : V$.

- $\nu \in \mathcal{X}$. We have $\nu \rho \in \llbracket V \rrbracket$, since $\rho$ is computable.

- $\nu = us$. We have $u : S \rightarrow V$ and $s : S$. By IH, $u \rho \in \llbracket S \rightarrow T \rrbracket$ and $s \rho \in \llbracket S \rrbracket$. Thus, $\nu \rho \in \llbracket V \rrbracket$. 
Abstraction case

\[ \nu = \lambda x.t. \] Let \( s_0 = \nu \rho, S_1 = \tau_x \) and assume that 
\[ \tau_t = S_2 \rightarrow \ldots \rightarrow S_n \rightarrow B \in \mathcal{B}. \] Let \( s_1 \in \llbracket S_1 \rrbracket, \ldots, s_n \in \llbracket S_n \rrbracket. \)

Possible reducts of \( s_0s_1\ldots s_n: \)

\[ t \rho_x^{s_1} s_2 \ldots s_n \in SN \text{ by IH} \]

\[ s_0 \ldots s_i' \ldots s_n \text{ with } s_i' \text{ a reduct of } s_i \]

Is every possible reduct \( SN \)?

Since each \( s_i \in SN \), the second case can be dealt with by well-founded induction on \((s_0, \ldots, s_n)\) if computability is preserved by reduction (the IH applies only if \( s_i' \) is computable).
Computability is preserved by reduction

**Lemma:** If \( v \in \llbracket V \rrbracket \) and \( v \rightarrow^\beta v' \), then \( v' \in \llbracket V \rrbracket \).

**Proof.** By induction on \( V \).

- **\( V \in B \).** Then, \( v \in SN \) and \( v' \in SN \). Thus, \( v' \in \llbracket V \rrbracket \).
- **\( V = S \rightarrow T \).** Let \( s \in \llbracket S \rrbracket \). Then, \( vs \in \llbracket T \rrbracket \). Since \( vs \rightarrow^\beta v's \), by IH, \( v's \in \llbracket T \rrbracket \). Thus, \( v' \in \llbracket V \rrbracket \).
**Final proof**

**Lemma:** If $v : V$ and $\rho$ is computable, then $v\rho \in [V]$.

Proof. By induction 1 on $v : V$.

- $v \in X$. We have $v\rho \in [V]$, since $\rho$ is computable.

- $v = us$. We have $u : S \to V$ and $s : S$. By IH, $u\rho \in [S \to T]$ and $s\rho \in [S]$. Thus, $v\rho \in [V]$.

- $v = \lambda x.t$. Let $s_0 = v\rho$, $S_1 = \tau_x$ and assume that $\tau_t = S_2 \to \ldots \to S_n \to B$. Let $s_1 \in [S_1], \ldots, s_n \in [S_n]$. We then prove that $s_0 s_1 \ldots s_n \in SN$ by well-founded induction 2 on $(s_0, \ldots, s_n)$. Possible reducts:
  - $t\rho_{\times}^{s_1} s_2 \ldots s_n$ is $SN$ by IH1.
  - $s_0 \ldots s'_i \ldots s_n$ with $s'_i$ a reduct of $s_i$ is $SN$ by IH2.
Consequences

**Lemma:** If $v : V$ and $\rho$ is computable, then $v\rho \in \llbracket V \rrbracket$.

**Corollary:** If $v : V$, then $v \in SN$.

**Proof.** Since $\llbracket V \rrbracket \subseteq SN$ and the identity substitution is computable.

**Corollary:** Every simply-typed $\lambda$-term has a unique $\beta$-normal form.

**Proof.** By termination, every term has at least one normal form. By confluence, every term has at most one normal form.

**Corollary:** $\beta$-equivalence is decidable.

**Proof.** Check that the $\beta$-normal forms are $\alpha$-equivalent.
What if we add constants and $\delta$-rules?

Take for instance the constants:

- $c : (T \to T) \to T$
- $p : T \to (T \to T)$

and the $\delta$-rule:

- $p(cx) \to x$

Do well-typed terms using $p$ and $c$ terminate?

Let $\omega = \lambda x^T.pxx : T \to T$.
Then, $\omega(c\omega) \to_\beta p(c\omega)(c\omega) \to_\delta w(c\omega) \to_\beta \ldots$!

Constants and rules introduce relations on types:

- $p$ maps every element of $T$ to a map from $T$ to $T$. Ok.
- $c$ maps every map from $T$ to $T$ to an element of $T$. Strange.
- $p(cx) \to x$ means that $T$ is in bijection with the set of functions from $T$ to $T$! This is possible only if $T = \emptyset$ (Cantor theorem).