Computability Closure: the Swiss knife of higher-order termination

Frédéric Blanqui

Inria
INVENTEURS DU MONDE NUMÉRIQUE

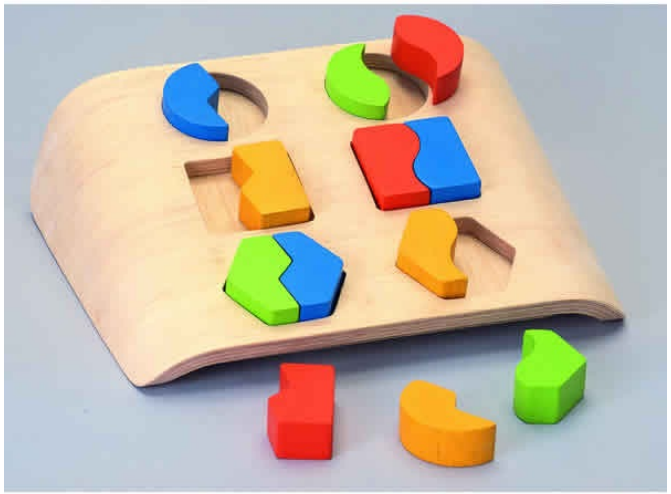
28 May 2012

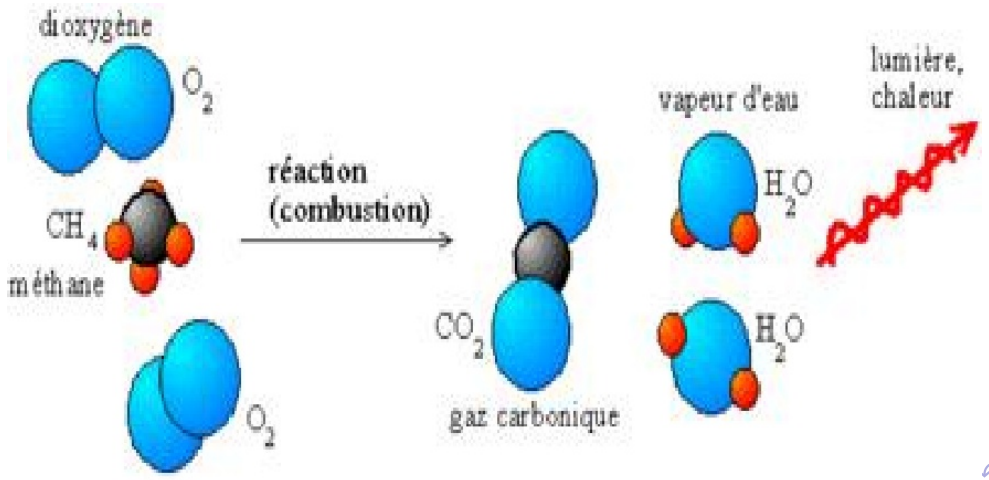
Rewriting

rewriting is a simple yet general framework for defining functions and proving equalities on terms based on the following notions:

- a rewrite rule is a pair of terms $l \to r$
- a substitution σ is a map from variables to terms
- a term t matches another term l if $t = l\sigma$

$$t \to_{\mathcal{R}} u \text{ if } \exists p, \exists \sigma, \exists l \to r \in \mathcal{R},\ t|_p = l\sigma \text{ and } u = t[r\sigma]_p$$





Higher-order rewriting

higher-order rewriting is rewriting on λ-terms (Church 1940)

$$\mathsf{f} \mid x \mid \lambda x t \mid tu$$

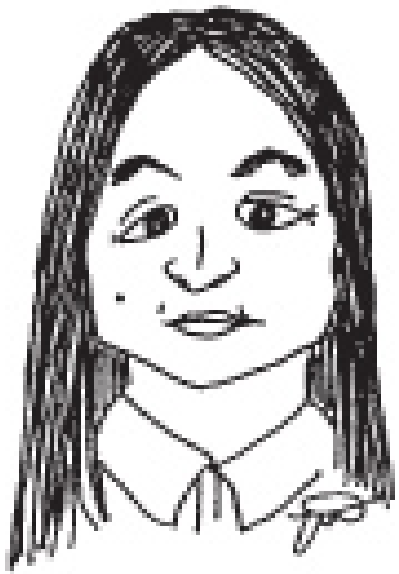
there are various approaches:

- Combinatory Reduction Systems (CRS) (Klop 1980)
- Expression Reduction Systems (ERS) (Khasidashvili 1990)
- Higher-order Rewrite Systems (HRS) (Nipkow 1991)
  - simply-typed λ-terms in β-normal η-long form
  - matching modulo $\alpha\beta\eta$



Higher-order rewriting

- Higher-order Algebraic Specification Languages (HOASL) (Jouannaud-Okada 1991)
  - arbitrary terms
  - matching modulo α



Problem

how to prove the termination of $\to_{\mathcal{R}}$ or $\to_\beta \cup \to_{\mathcal{R}}$?

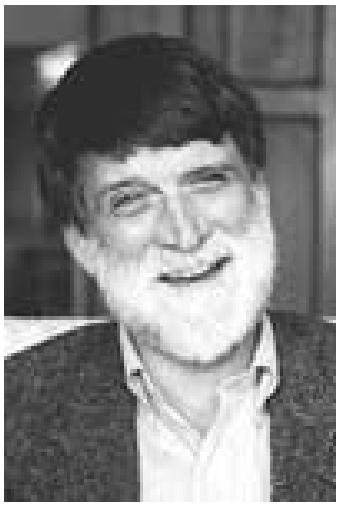
in the simply-typed λ-calculus:

- $\to_\beta$ can be proved terminating by a direct induction on the type of the substituted variable (Sanchis 1967, van Daalen 1980)
  this does not extend to rewriting since, in this case, the type of substituted variables can increase
- λI-terms can be interpreted by hereditarily monotone functions on $\mathbb{N}$ (Gandy 1980)
  this can be used to build interpretations (van de Pol 1996, Hamana 2006) but these interpretations can also be obtained from an extended computability proof

Outline

Computability

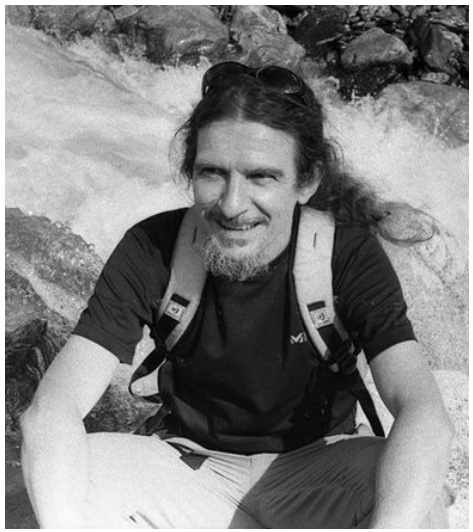
computability has been introduced for proving termination of β-reduction, *i.e.* substitution, in typed λ-calculi by William Walker Tait (1967) and Jean-Yves Girard (1970)



- every type T is mapped to a set $[\![T]\!]$ of computable terms
- every $t : T$ is proved to be computable, *i.e.* $t \in [\![T]\!]$

Computability predicates

there are different definitions of computability (Tait, Girard, Parigot) but Girard's definition Red is better suited for arbitrary rewriting



Let Red be the set of P such that:

- $P \subseteq \mathrm{SN}(\to_\beta)$
- $\to_\beta (P) \subseteq P$
- if t is neutral and $\to_\beta (t) \subseteq P$ then $t \in P$

t cannot be head-reduced when applied (*e.g.* $\lambda x u$ is not neutral)

Computable terms

Red is a complete lattice for set inclusion closed by:

$$a(P, Q) = \{t \mid \forall u \in P, tu \in Q\}$$

by taking $[\![U \Rightarrow V]\!] := a([\![U]\!], [\![V]\!])$, a term $t : U \Rightarrow V$ is computable if, for every computable $u : U$, tu is computable

Application to rewriting (Jouannaud-Okada 1991)

Given a set $\mathcal{R}$ of rewrite rules, let $\rightarrow = \rightarrow_\beta \cup \rightarrow_\mathcal{R}$ and $\mathrm{Red}_\mathcal{R}$ be the set of P such that:

- $P \subseteq \mathrm{SN}(\rightarrow)$
- $\rightarrow(P) \subseteq P$
- if t is neutral and $\rightarrow(t) \subseteq P$ then $t \in P$
  $\mathrm{f}\vec{t}$ is neutral if $|\vec{t}| \geq \sup\{|\vec{l}| \mid \mathrm{f}\vec{l} \rightarrow r \in \mathcal{R}\}$

Theorem: Given a set $\mathcal{R}$ of rules, the relation $\rightarrow_\beta \cup \rightarrow_\mathcal{R}$ terminates if every rule of $\mathcal{R}$ is of the form $\mathrm{f}\vec{l} \rightarrow r$ with $r \in \mathrm{CC}_{\mathcal{R},\mathrm{f}}(\vec{l})$, where $\mathrm{CC}_{\mathcal{R},\mathrm{f}}(\vec{l})$ is a set of terms $\mathcal{R}$-computable whenever $\vec{l}$ so are.

Computability closure

By what operation $\mathrm{CC}_{\mathcal{R},\mathrm{f}}(\vec{l})$ can be closed?

$$\text{(arg)}\ l_i \in \mathrm{CC}_{\mathcal{R},\mathrm{f}}(\vec{l})$$

$$\text{(app)}\ \frac{t : U \Rightarrow V \in \mathrm{CC}_{\mathcal{R},\mathrm{f}}(\vec{l}) \quad u : U \in \mathrm{CC}_{\mathcal{R},\mathrm{f}}(\vec{l})}{tu \in \mathrm{CC}_{\mathcal{R},\mathrm{f}}(\vec{l})}$$

$$\text{(red)}\ \frac{t \in \mathrm{CC}_{\mathcal{R},\mathrm{f}}(\vec{l}) \quad t \to_\beta \cup \to_{\mathcal{R}} t'}{t' \in \mathrm{CC}_{\mathcal{R},\mathrm{f}}(\vec{l})}$$

Dealing with bound variables

Annotate $\mathrm{CC}_{\mathcal{R},\mathsf{f}}(\vec{l})$ with a set X of (bound) variables:

$$\text{(var)}\ \frac{x \in X}{x \in \mathrm{CC}^{X}_{\mathcal{R},\mathsf{f}}(\vec{l})}$$

$$\text{(lam)}\ \frac{t \in \mathrm{CC}^{X \cup \{x\}}_{\mathcal{R},\mathsf{f}}(\vec{l}) \quad x \notin \mathrm{FV}(\vec{l})}{\lambda x t \in \mathrm{CC}^{X}_{\mathcal{R},\mathsf{f}}(\vec{l})}$$

Dealing with subterms

Problem: computability is not preserved by subterm. . . :-(

Example: with $\mathsf{c} : (\mathsf{B} \Rightarrow \mathsf{A}) \Rightarrow \mathsf{B}$ and $\mathsf{f} : \mathsf{B} \Rightarrow (\mathsf{B} \Rightarrow \mathsf{A})$, $\rightarrow_\beta \cup \rightarrow_{\mathcal{R}}$ with $\mathcal{R} = \{\mathsf{f}(\mathsf{c}x) \rightarrow x\}$ does not terminate (Mendler 1987)

with $w = \lambda x\mathsf{f}xx : \mathsf{B} \Rightarrow \mathsf{A}$, $w(\mathsf{c}w) \rightarrow_\beta \mathsf{f}(\mathsf{c}w)(\mathsf{c}w) \rightarrow_{\mathcal{R}} w(\mathsf{c}w)$

$\Rightarrow$ restrictions on subterms (based on types) are necessary:

$$\text{(sub-app-fun)}\quad \frac{\mathsf{g}\vec{t} \in \mathrm{CC}^{\mathsf{X}}_{\mathcal{R},\mathsf{f}}(\vec{l}) \quad \mathsf{g} : \vec{T} \Rightarrow \mathsf{B} \quad \mathrm{Pos}(\mathsf{B}, T_i) \subseteq \mathrm{Pos}^+(T_i)}{t_i \in \mathrm{CC}^{\mathsf{X}}_{\mathcal{R},\mathsf{f}}(\vec{l})}$$

Dealing with subterms

$$\text{(sub-app-var-l)}\ \frac{tu \in \mathrm{CC}^X_{\mathcal{R},\mathsf{f}}(\vec{l}) \quad u{\downarrow_\eta} \in X}{t \in CC^X_{\mathsf{f}}(\vec{l})}$$

$$\text{(sub-app-var-r)}\ \frac{tu \in \mathrm{CC}^X_{\mathcal{R},\mathsf{f}}(\vec{l}) \quad t{\downarrow_\eta} \in X \quad t : U \Rightarrow \vec{U} \Rightarrow U}{u \in CC^X_{\mathsf{f}}(\vec{l})}$$

$$\text{(sub-lam)}\ \frac{\lambda x t \in \mathrm{CC}^X_{\mathcal{R},\mathsf{f}}(\vec{l}) \quad x \notin \mathrm{FV}(\vec{l})}{t \in \mathrm{CC}^{X\cup\{x\}}_{\mathcal{R},\mathsf{f}}(\vec{l})}$$

$$\text{(sub-SN)}\ \frac{t \in \mathrm{CC}^X_{\mathcal{R},\mathsf{f}}(\vec{l}) \quad u : \mathsf{B} \trianglelefteq t \quad \mathrm{FV}(u) \subseteq \mathrm{FV}(t) \quad [\![\mathsf{B}]\!] = \mathrm{SN}}{u \in \mathrm{CC}^X_{\mathcal{R},\mathsf{f}}(\vec{l})}$$

Dealing with function calls

Consider a relation $\sqsupset$ on pairs $(\mathsf{h}, \vec{v})$, where $\vec{v}$ are computable arguments of h, such that $\sqsupset \cup \to_{\mathrm{prod}}$ is well-founded.

$$\text{(app-fun)}\ \frac{(\mathsf{f}, \vec{l}) \sqsupset (\mathsf{g}, \vec{t}) \quad \vec{t} \in \mathrm{CC}_{\mathcal{R},\mathsf{f}}(\vec{l})}{\mathsf{g}\vec{t} \in \mathrm{CC}_{\mathcal{R},\mathsf{f}}(\vec{l})}$$

Example: $(\mathsf{f}, \vec{l}) \sqsupset (\mathsf{g}, \vec{t})$ if either:

- $\mathsf{f} > \mathsf{g}$
- $\mathsf{f} \simeq \mathsf{g}$ and $\vec{l}\ ((\rhd \cup \to)^+)_{\mathrm{stat}[\mathsf{f}]}\ \vec{t}$

where $\geq$ is a well-founded quasi-ordering on symbols and $\mathrm{stat}[\mathsf{f}] = \mathrm{stat}[\mathsf{g}] \in \{\mathrm{lex}, \mathrm{mul}\}$

Outline

Dealing with higher-order pattern-matching

$$\mathsf{f}\vec{t} =_{\beta_0\eta} \mathsf{f}\vec{l}\sigma \to_{\mathcal{R}} r\sigma$$

Problem: $\vec{t}$ computable $\Rightarrow$ $\vec{l}\sigma$ computable?

Dealing with higher-order pattern-matching

Dale Miller (1991): if l is an *higher-order pattern* and $l\sigma =_{\beta\eta} t$ with σ and t in β-normal η-long form, then $l\sigma \to^*_{\beta_0} =_\eta t$ where $C[(\lambda x u)v] \to_{\beta_0} C[u^v_x]$ if $v \in \mathcal{X}$

$\Rightarrow$ consider β_0-normalized rewriting with matching modulo $\beta_0\eta$ (subsumes CRS and HRS rewriting)!

Theorem: assuming that $\leftarrow_{\beta_0\eta} \to_{\mathcal{R},\beta_0\eta} \subseteq \to_{\mathcal{R},\beta_0\eta} =_{\beta_0\eta}$, if t is computable and $t =_{\beta_0\eta} l\sigma$ with l an higher-order pattern, then $l\sigma$ is computable.

Dealing with higher-order pattern-matching

Theorem: $\leftarrow_{\beta_0\eta} \rightarrow_{\mathcal{R},\beta_0\eta} \subseteq \rightarrow_{\mathcal{R},\beta_0\eta =_{\beta_0\eta}}$ if:

- every rule is of the form $f\vec{l} \rightarrow r$ with $f\vec{l}$ an higher-order pattern
- if $l \rightarrow r \in \mathcal{R}$, $l : T \Rightarrow U$ and $x \notin \mathrm{FV}(l)$, then $lx \rightarrow rx \in \mathcal{R}$
- if $lx \rightarrow r \in \mathcal{R}$ and $x \notin \mathrm{FV}(l)$, then $l \rightarrow \lambda x r \in \mathcal{R}$

$$s \leftarrow_{\beta_0} (\lambda x s)x =_{\beta_0\eta} l\sigma x \rightarrow_{\mathcal{R}} r\sigma x$$

$$s \leftarrow_{\eta} \lambda x s x =_{\beta_0\eta} \lambda x l\sigma \rightarrow_{\mathcal{R}} \lambda x r\sigma$$

$\Rightarrow$ every set of rules of the form $f\vec{l} \rightarrow r$ with $f\vec{l}$ an higher-order pattern can be completed into a set compatible with $\rightarrow_{\beta_0\eta}$

Outline

Dealing with rewriting modulo some equational theory

$$f\vec{t} =_{\mathcal{E}} u \to_{\mathcal{R}} v$$

Problem: $\vec{t}$ computable $\Rightarrow$ v computable?

Dealing with rewriting modulo some equational theory

First, we need $\mathrm{SN}(\to_\beta)$ to be closed by $=_\mathcal{E}$. For instance:

Theorem: $\to_\beta{=_\mathcal{E}} \subseteq {=_\mathcal{E}}\to_\beta$ if:

- $\mathcal{E}$ is linear
- $\mathcal{E}$ is regular $(\forall l = r \in \mathcal{E}, \mathrm{FV}(l) = \mathrm{FV}(r))$
- $\mathcal{E}$ is algebraic (no abstraction nor applied variable)

 $x \times 0 = 0$

 $x \times (y + z) = (x \times y) + (x \times z)$

 $\forall(\lambda x \forall(\lambda y P x y)) = \forall(\lambda y \forall(\lambda x P x y))$

Dealing with rewriting modulo some equational theory

Given a set $\mathcal{E}$ of equations and a set $\mathcal{R}$ of rewrite rules, let now $\rightarrow = \rightarrow_\beta \cup =_\mathcal{E} \rightarrow_\mathcal{R}$ and $\mathrm{Red}^\mathcal{E}_\mathcal{R}$ be the set of P such that:

- $P \subseteq \mathrm{SN}(\rightarrow)$
- $\rightarrow(P) \subseteq P$ and $=_\mathcal{E}(P) \subseteq P$
- if t is neutral and $\rightarrow(t) \subseteq P$ then $t \in P$

Dealing with rewriting modulo some equational theory

Theorem: assuming that $\to_\beta{=_\mathcal{E}} \subseteq {=_\mathcal{E}}{\to_\beta}$, the relation $\to_\beta \cup {=_\mathcal{E}}{\to_\mathcal{R}}$ terminates if:

- every rule of $\mathcal{R}$ is of the form $\mathsf{h}\vec{n} \to r$ with $r \in \mathrm{CC}^{\mathcal{E}}_{\mathcal{R},\mathsf{h}}(\vec{n})$,
- every equation of $\mathcal{E}$ is of the form $\mathsf{f}\vec{l} = \mathsf{g}\vec{m}$ with $\vec{m} \in \mathrm{CC}^{\mathcal{E}}_{\mathcal{R},\mathsf{f}}(\vec{l})$ and $\vec{l} \in \mathrm{CC}^{\mathcal{E}}_{\mathcal{R},\mathsf{g}}(\vec{m})$.

$$\mathsf{f}\vec{t} = \mathsf{f}\vec{l}\sigma \leftrightarrow_\mathcal{E} \mathsf{g}\vec{m}\sigma \leftrightarrow_\mathcal{E} \ldots \leftrightarrow_\mathcal{E} \mathsf{h}\vec{n}\theta \to_\mathcal{R} r\theta = v$$

$$\vec{t} \text{ computable} \Rightarrow \vec{m}\sigma \text{ computable} \Rightarrow \ldots \Rightarrow v \text{ computable}$$

Dealing with rewriting modulo some equational theory

Examples:

- commutativity: $+xy = +yx$

$$\{y, x\} \subseteq \mathrm{CC}_+(xy)$$

- associativity: $+(+xy)z = +x(+yz)$

$$\{x, +yz\} \subseteq \mathrm{CC}_+((+xy)z)$$
$$\{+xy, z\} \subseteq \mathrm{CC}_+(x(+yz))$$

Outline

RPO

RPO is a well-founded quasi-ordering on FO terms extending a well-founded quasi-ordering $>$ on symbols (Plaisted-Dershowitz 78)

$$(1)\ \frac{t_i \geq_{\mathrm{rpo}} u}{\mathsf{f}\vec{t} >_{\mathrm{rpo}} u} \qquad (2)\ \frac{(\mathsf{f},\vec{t}) \sqsupset (\mathsf{g},\vec{u}) \quad \mathsf{f}\vec{t} >_{\mathrm{rpo}} \vec{u}}{\mathsf{f}\vec{t} >_{\mathrm{rpo}} \mathsf{g}\vec{u}}$$

where $(\mathsf{f},\vec{t}) \sqsupset (\mathsf{g},\vec{u})$ if either $\mathsf{f} > \mathsf{g}$ or $\mathsf{f} \simeq \mathsf{g}$ and $\vec{t}\ (>_{\mathrm{rpo}})_{\mathrm{stat}[\mathsf{f}]}\ \vec{u}$

HORPO

HORPO is a non-transitive extension of RPO to λ-terms (Jouannaud-Rubio 99)

Revisiting (HO)RPO

What is the relation between CC and HORPO?

- both are based on computability
- there are even extensions of HORPO using CC
- CC is defined for a fixed $\mathcal{R}$

CC is itself a relation!

replace $t \in \mathrm{CC}_{\mathcal{R},\mathsf{f}}(\vec{l})$ by $\mathsf{f}\vec{l} >_{\mathrm{CC}(\mathcal{R})} t$

Revisiting (HO)RPO

$$\text{(arg)}\ \mathsf{f}\vec{l} >_{\mathrm{CC}(\mathcal{R})} l_i$$

$$\text{(red)}\ \frac{\mathsf{f}\vec{l} >_{\mathrm{CC}(\mathcal{R})} t \quad t \to_\beta \cup \to_{\mathcal{R}} t'}{\mathsf{f}\vec{l} >_{\mathrm{CC}(\mathcal{R})} t'}$$

$$\text{(app-fun)}\ \frac{(\mathsf{f}, \vec{l}) \sqsupset (\mathsf{g}, \vec{t}) \quad \mathsf{f}\vec{l} >_{\mathrm{CC}(\mathcal{R})} \vec{t}}{\mathsf{f}\vec{l} >_{\mathrm{CC}(\mathcal{R})} \mathsf{g}\vec{t}}$$

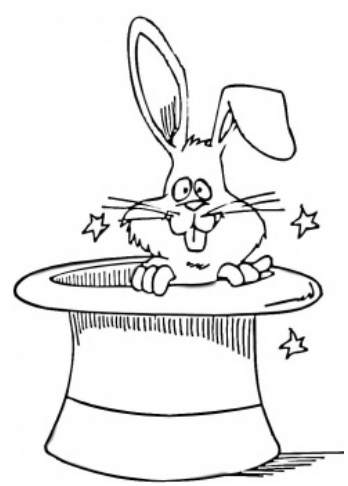
$$(\mathsf{f}, \vec{l}) \sqsupset (\mathsf{g}, \vec{t}) \text{ if either } \mathsf{f} > \mathsf{g}$$
$$\text{or } \mathsf{f} \simeq \mathsf{g} \text{ and } \vec{l}\ ((\rhd \cup \to_\beta \cup \to_{\mathcal{R}})^+)_{\mathrm{stat}[\mathsf{f}]}\ \vec{t}$$

...

Revisiting (HO)RPO

$$\mathcal{R} \mapsto \{(\mathrm{f}\vec{l}, r) \mid r \in \mathrm{CC}^{\emptyset}_{\mathcal{R},\mathrm{f}}, \mathrm{type}(\mathrm{f}\vec{l}) = \mathrm{type}(r)\}$$

is a monotone function on the complete lattice of relations



the monotone closure of its least fixpoint:

- contains HORPO
- is equal to RPO when restricted to FO terms!

$\Rightarrow$ this provides a general method to easily get a powerful ordering for richer type systems

To know more on computability closure

- how to deal with constructors with functional arguments
- how to deal with conditional rewriting
- what is the relation with dependency pairs
- what is the relation with semantic labelling

see `https://who.rocq.inria.fr/Frederic.Blanqui/`

Thank you!