Confluence and strong normalization for the CPS target language*

Jonas Frey

May 1, 2016

Abstract

We prove confluence and strong normalization for the *CPS target language*, a higher order calclus that is closely related to Girard's ludics [Gir01] and was introduced in the present form by the author [Fre16] as part of a reformulation of Krivine's classical realizability [Kri09].

Confluence is shown for the untyped language, and strong normalization for a Curry-style type system with second order quantifiers.

1 The CPS target language

The syntax of the CPS target language, given in Table 1, distinguishes two syntactic classes called *terms* and *programs*.

A *term* is either a variable or a *record*, i.e. a family $\langle \ell_1(y.\, p_1), \dots, \ell_m(y.\, p_m)\rangle$ of programs p_i indexed by a finite subset $\{\ell_1, \dots, \ell_m\} \subseteq \mathscr{L}$ of a countable set of *labels*. The programs in a record – called its *methods* – are each abstracted by a designated variable y.

A *program* is an expression of the form $t_\ell u$, with the intended meaning that the method of t with label ℓ is called with u as an argument. This reading suggests the reduction rule $\langle \dots, \ell(y.\, p), \dots\rangle_\ell t \succ p[t/y]$, which gives the operational semantics of the language. We use the symbol $\succ$ only for top-level reduction of programs (i.e. *weak head reduction*), and write $\to_\beta$ for its *compatible closure* on terms and programs, which explicitly is inductively defined by the following clauses.

- $\langle \dots, \ell(y.\, p), \dots\rangle_\ell t \to_\beta p[t/y]$
- If $p \to_\beta p'$ then $\langle \dots, \ell(y.\, p), \dots\rangle \to_\beta \langle \dots, \ell(y.\, p'), \dots\rangle$
- If $t \to_\beta t'$ then $t_\ell u \to_\beta t'_\ell u$ and $u_\ell t \to_\beta u_\ell t'$

A *redex* is a program $t_\ell u$ where t is a record. A redex $\langle \ell_1(x.\, p_1), \dots, \ell_m(x.\, p_m)\rangle_\Bbbk t$ with $\Bbbk \notin \{\ell_1, \dots, \ell_m\}$ cannot be reduced and is said to be *blocked*. A *normal form* is a term or program that does not contain any redexes, i.e. in every application $t_\ell u$ the term t is a variable. A term or program is said to be *strongly normalizing*, if every $\to_\beta$-reduction sequence leads to a normal form.

We define the sets $\mathrm{FV}(t)$ and $\mathrm{FV}(p)$ of *free variables* of a term or program in the usual way, where the distinguished variable y of a method $\ell(y.\, p)$ in a record t is considered bound

*This work is supported by the Danish Council for Independent Research *Sapere Aude* grant "Complexity via Logic and Algebra" (COLA).

Expressions:

$$\begin{array}{llll} \textit{Terms:} & s,t,u & ::= & x \mid \langle \ell_1(y.\,p_1),\dots,\ell_n(y.\,p_m)\rangle \\ \textit{Programs:} & p,q & ::= & t_\ell u \end{array}$$

Reduction:

$$\langle \dots, \ell(y.\,p), \dots \rangle_\ell t \quad \succ \quad p[t/y]$$

Types:

$$A \quad ::= \quad X \mid \left\langle \ell_1(A_1),\dots,\ell_n(A_m)\right\rangle \mid \forall X A \qquad n \geq 0$$

Typing rules:

$$\text{(Var)} \quad \frac{}{\Gamma \vdash x_i : A_i} \qquad \Gamma \equiv x_1 : A_1, \dots, x_n : A_n,\ \ 1 \leq i \leq n$$

$$\text{(Abs)} \quad \frac{\Gamma, y : B_1 \vdash p_1 \qquad \cdots \qquad \Gamma, y : B_m \vdash p_m}{\Gamma \vdash \langle \ell_1(y.\,p_1),\dots,\ell_m(y.\,p_m)\rangle : \left\langle \ell_1(B_1),\dots,\ell_m(B_m)\right\rangle}$$

$$\text{(App)} \quad \frac{\Gamma \vdash t : \left\langle \ell_1(B_1),\dots,\ell_m(B_m)\right\rangle \qquad \Gamma \vdash u : B_i}{\Gamma \vdash t_{\ell_i} u} \quad 1 \leq i \leq m$$

$$\text{(Gen)} \quad \frac{\Gamma \ \vdash \ t : A}{\Gamma \ \vdash \ t : \forall X A} \quad X \notin \mathrm{FV}(\Gamma)$$

$$\text{(Inst)} \quad \frac{\Gamma \ \vdash \ t : \forall X A}{\Gamma \ \vdash \ t : A[B/X]}$$

Table 1: The CPS target language.

in p. There are no closed normal programs (since the term in head position cannot be a variable) but there are blocked closed programs like $\langle\rangle_{\mathsf{k}}\langle\rangle$ and diverging closed programs like $\langle \mathsf{k}(x.\,x_{\mathsf{k}}x)\rangle_{\mathsf{k}}\langle \mathsf{k}(x.\,x_{\mathsf{k}}x)\rangle$.

We denote the sets of terms and programs by $\mathbb{T}$ and $\mathbb{P}$, and we denote the sets of strongly normalizing terms and programs by $\mathbb{SNT} \subseteq \mathbb{T}$ and $\mathbb{SNP} \subseteq \mathbb{P}$ (this notation deviates a bit from [Fre16], where $\mathbb{T}$ and $\mathbb{P}$ denote the sets of *closed* terms and programs).

We consider a Curry-style type system for the CPS target language, whose types are generated from type variables X, second order quantifiers $\forall X$, and for each finite set $\{\ell_1,\dots,\ell_n\}$ of labels an n-ary type constructor which forms the record type $\left\langle \ell_1(A_1),\dots,\ell_n(A_n)\right\rangle$ out of types $A_1,\dots,A_n$. There are two kinds of typing judgments corresponding to the two syntactic classes:

- Terms t are typed by sequents $x_1 : A_1 \dots x_n : A_n \vdash t : B$, where $\mathrm{FV}(t) \subseteq \{x_1,\dots,x_n\}$.
- Programs p are typed by sequents $x_1 : A_1 \dots x_n : A_n \vdash p$, where $\mathrm{FV}(p) \subseteq \{x_1,\dots,x_n\}$.

Thus, programs are not associated to types, but we think of them as having type $\bot$. There are five typing rules: a variable rule (Var), abstraction and application rules (Abs) and (App) to type records and programs, and introduction and elimination rules (Gen) and (Inst) for second order quantification.

2 Confluence

To prove confluence of the reduction relation $\rightarrow_\beta$ on the untyped language we give a straightforward adaptation of the confluence proof of untyped λ-calculus using *parallel reduction* and *complete developments*, as presented in Sørensen and Urzyczyn's textbook [SU06, Section 1.4].

Definition 2.1 The *parallel reduction* relation $\Rightarrow_\beta$ is defined as follows by simultaneous induction on the structure of terms and programs of the CPS target language.

- $x \Rightarrow_\beta x$ for all variables x.
- If $p_i \Rightarrow_\beta p_i'$ for $1 \leq i \leq m$ then $\langle \ell_1(y.\, p_1), \ldots, \ell_m(y.\, p_m)\rangle \Rightarrow_\beta \langle \ell_1(y.\, p_1'), \ldots, \ell_m(y.\, p_m')\rangle$.
- If $s \Rightarrow_\beta s'$ and $t \Rightarrow_\beta t'$ then $s_{\not k} t \Rightarrow_\beta s'_{\not k} t'$.
- If $p_i \Rightarrow_\beta p_i'$ and $t \Rightarrow_\beta t'$ then $\langle \ell_1(y.\, p_1), \ldots, \ell_m(y.\, p_m)\rangle_{\ell_i} t \Rightarrow_\beta p_i'[t'/y]$. ♢

Lemma 2.2

1. *$t \Rightarrow_\beta t$ and $p \Rightarrow_\beta p$ for all terms t and programs p.*
2. *If $t \rightarrow_\beta t'$ then $t \Rightarrow_\beta t'$. If $p \rightarrow_\beta p'$ then $p \Rightarrow_\beta p'$.*
3. *If $t \Rightarrow_\beta t'$ then $t \rightarrow_\beta^* t'$. If $p \Rightarrow_\beta p'$ then $p \rightarrow_\beta^* p'$.*
4. *If $t \Rightarrow_\beta t'$ and $u \Rightarrow_\beta u'$ then $t[u/x] \Rightarrow_\beta t'[u'/x]$.*
   If $p \Rightarrow_\beta p'$ and $u \Rightarrow_\beta u'$ then $p[u/x] \Rightarrow_\beta p'[u'/x]$.

Proof. Easy inductions. ■

Definition 2.3 The *complete developments* t^* and p^* of terms t and programs p are defined as follows.

$$
\begin{array}{lcll}
x^* & = & x & \\
\langle \ell_1(y.\, p_1), \ldots, \ell_n(y.\, p_m)\rangle^* & = & \langle \ell_1(y.\, p_1^*), \ldots, \ell_n(y.\, p_m^*)\rangle & \\
(x_\ell t)^* & = & x_{\not k} t^* & \\
(\langle \ell_1(y.\, p_1), \ldots, \ell_n(y.\, p_m)\rangle_{\not k} t)^* & = & \langle \ell_1(y.\, p_1^*), \ldots, \ell_n(y.\, p_m^*)\rangle_{\not k} t^* & \text{if } \not k \notin \{\ell_1, \ldots, \ell_m\} \\
(\langle \ldots, \ell(y.\, p), \ldots\rangle_\ell t)^* & = & p^*[t^*/y] &
\end{array}
$$

Lemma 2.4 *If $t \Rightarrow_\beta t'$ then $t' \Rightarrow_\beta t^*$. If $p \Rightarrow_\beta p'$ then $p' \Rightarrow_\beta p^*$.*

Proof. Induction on the derivation of $t \Rightarrow_\beta t'$ and $p \Rightarrow_\beta p'$. ■

Now given an expression (i.e. a term or a program) e and parallel reductions $e \Rightarrow_\beta e'$ and $e \Rightarrow_\beta e''$, the preceding lemma implies that $e' \Rightarrow_\beta e^*$ and $e'' \Rightarrow_\beta e^*$, which means that $\Rightarrow_\beta$ has the *diamond property* [BKdV03, Def. 1.1.8-(v)], and is therefore confluent. Since $(\rightarrow_\beta^*) = (\Rightarrow_\beta^*)$ by Lemma 2.2, $\rightarrow_\beta$ is confluent as well.

3 Strong normalization

In this section we show that all typable terms and programs of the CPS target language are strongly normalizing. The proof is an adaptation of Girard's proof [Gir72] of strong normalization for System F using *reducibility candidates*; our primary reference is again Sørensen and Urzyczyn's textbook. The principal difference is that in the case of the CPS target language the expressions are divided into two syntactic classes, which simplifies the proof.

As in the classical proof we define a *candidate* to be a set of terms satisfying certain conditions (compare [SU06, Def. 11.5.1]), but in our case the condition involving head reduction does not apply since head reduction is defined on programs, not terms.

Definition 3.1 A set $S \subseteq \mathbb{T}$ of terms is called a *candidate* if it contains all variables and all its elements are strongly normalizing. We denote the set of candidates by $\mathscr{C}$. ♢

Given a type A and and a valuation ξ assigning candidates to the free type variables of A, the set $[\![A]\!]_\xi \subseteq \mathbb{T}$ is defined as follows by induction on the type structure.

1. $[\![X]\!]_\xi = \xi(X)$
2. $[\![\langle\rangle]\!]_\xi = \mathbb{SNT}$
3. $[\![\langle \ell_1(A_1), \ldots, \ell_n(A_n)\rangle]\!]_\xi = \bigcap_{1\le i\le n}\{t \in \mathbb{T} \mid \forall s \in [\![A_i]\!]_\xi \,.\, t_{\ell_i} s \in \mathbb{SNP}\}$ for $n \ge 1$
4. $[\![\forall X\, A]\!]_\xi = \bigcap_{S\in\mathscr{C}} [\![A]\!]_{\xi\{X\mapsto S\}}$

The second and third clause can be combined if we accept that $\mathbb{SNT}$ is the empty intersection, and in the last clause $\xi\{X \mapsto S\}$ is the extension of ξ that maps X to the candidate S. Now the technical centerpiece of the proof are the following lemma and theorem.

Lemma 3.2 *$[\![A]\!]_\xi$ is a candidate for all types A and valuations ξ.*

Proof. By induction on the structure of A. 1, 2, and 4 are immediate. For 3 we show that $\{t \in \mathbb{T} \mid \forall s \in [\![A_i]\!]_\xi \,.\, t_{\ell_i} s \in \mathbb{SNP}\}$ is a candidate for fixed i. We see that x is contained since $x_{\ell_i} s$ normalizes strongly whenever s does, and any element $t \in \{t \in \mathbb{T} \mid \forall s \in [\![A_i]\!]_\xi \,.\, t_{\ell_i} s \in \mathbb{SNP}\}$ normalizes strongly since $t_{\ell_i} x$ does. ■

Theorem 3.3 *Let $\Gamma \equiv x_1 : A_1, \ldots, x_n : A_n$ be a context and ξ a valuation.*

1. *Given a judgment $\Gamma \vdash t : B$ and terms $\vec{s} \in [\![\Gamma]\!]_\xi$[1], we have $t[\vec{s}/\vec{x}] \in [\![B]\!]_\xi$.*
2. *Given a judgment $\Gamma \vdash p$ and terms $\vec{s} \in [\![\Gamma]\!]_\xi$, we have $p[\vec{s}/\vec{x}] \in \mathbb{SNP}$.*

Proof. By simultaneous induction on the derivations of $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$ and $x_1 : A_1, \ldots, x_n : A_n \vdash p$. We only consider the rule (Abs) – which is most interesting – and leave the other cases to the reader.

In the 0-ary case we only have to show that the empty record $\langle\rangle$ is strongly normalizing, which is true since it is a normal form. For the non-empty case assume that the judgment

$$\Gamma \vdash \langle \ell_1(y.\, p_1), \ldots, \ell_m(y.\, p_m)\rangle : \langle \ell_1(B_1), \ldots, \ell_m(B_m)\rangle$$

[1] i.e. $\vec{s} = (s_1, \ldots, s_n)$ with $s_1 \in [\![A_1]\!]_\xi, \ldots, s_n \in [\![A_n]\!]_\xi$

is derived via (Abs) and that $\vec{s} \in [\![\Gamma]\!]_\xi$. We have to show that

$$u \;=\; \langle \ell_1(y.\, p_1[\vec{s}/\vec{x}]), \ldots, \ell_m(y.\, p_m[\vec{s}/\vec{x}])\rangle_{\ell_i} t$$

is strongly normalizing for all $1 \leq i \leq m$ and $t \in [\![B_i]\!]_\xi$. Note that the programs $p_j[\vec{s}/\vec{x}] = p_j[\vec{s}/\vec{x}, y/y]$ are all in $\mathbb{SNP}$ by hypothesis, since $(\vec{s}, y) \in [\![\Gamma, y : B_j]\!]_\xi$. Now assume that there is an infinite reduction starting from u. Since the $p_j[\vec{s}/\vec{x}]$ as well as t are strongly normalizing, this reduction does eventually have to reduce the head redex, i.e. $u \to_\beta^* q[v/y]$ with for reducts q of $p_i[\vec{s}/\vec{x}]$ and v of t. Now $q[v/y]$ is also a reduct of $p_i[\vec{s}/\vec{x}, t/y]$, which is strongly normalizing by hypothesis because $(\vec{s}, t) \in [\![\Gamma, y : B_i]\!]_\xi$. This contradicts the assumption of an infinite reduction sequence starting with u. For a similar reason no reduction of u can end in a blocked state, neither before nor after reduction of the head redex. ∎

Choosing ξ to be an arbitary valuation (e.g. the one assigning $\mathbb{SNT}$ to all type variables) and setting $\vec{s} = \vec{x}$ in the theorem, we obtain the following.

Corollary 3.4 *Typable terms and programs are strongly normalizing.* ∎

References

[BKdV03] M. Bezem, J.W. Klop, and R. de Vrijer. *Term rewriting systems*. Cambridge University Press, 2003.

[Fre16] J. Frey. Classical realizability in the CPS target language. 2016. Accepted at MFPS 2016, available at https://sites.google.com/site/jonasfreysite/mfps.pdf.

[Gir72] J.Y. Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur*. PhD thesis, Université Paris VII, 1972.

[Gir01] J.Y. Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical Structures in Computer Science*, 11(03):301–506, 2001.

[Kri09] J.L. Krivine. Realizability in classical logic. *Panoramas et synthèses*, 27:197–229, 2009.

[SU06] M. Sørensen and P. Urzyczyn. *Lectures on the Curry-Howard isomorphism*, volume 149. Elsevier, 2006.