Dependency Pairs Termination in Dependent Type Theory Modulo Rewriting

Frédéric Blanqui, Guillaume Genestier, Olivier Hermant

▶ **To cite this version:**

Frédéric Blanqui, Guillaume Genestier, Olivier Hermant. Dependency Pairs Termination in Dependent Type Theory Modulo Rewriting. TYPES 2019 - 25th International Conference on Types for Proofs and Programs, Jun 2019, Oslo, Norway. pp.30-31. hal-02442484

HAL Id: hal-02442484
https://hal.science/hal-02442484

Submitted on 16 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Dependency Pairs Termination in Dependent Type Theory Modulo Rewriting

Frédéric Blanqui[1,2], Guillaume Genestier[2,3], and Olivier Hermant[3]

[1] INRIA
[2] LSV, ENS Paris-Saclay, CNRS, Université Paris-Saclay
[3] MINES ParisTech, PSL University

We introduce a termination criterion for a large class of programs whose operational semantics can be described by higher-order rewriting rules typable in the $\lambda\Pi$-calculus modulo rewriting ($\lambda\Pi/\mathcal{R}$ for short).

$\lambda\Pi/\mathcal{R}$ is a system of dependent types where types are identified modulo the β-reduction of λ-calculus and rewriting rules given by the user to define not only functions but also types. Those rewriting rules can be non-orthogonal, meaning that they can overlap or be non-linear. An example including overlapping and a type defined by rewriting rules is the propositional integer comparison:

```
symbol Prop : TYPE                symbol ⊥ : Prop          symbol ⊤ : Prop
symbol Prf : Prop ⇒ TYPE          symbol infix ≤ : Nat ⇒ Nat ⇒ Prop
 rule Prf ⊤ ⟶ Πc:Prop. Prf c ⇒ Prf c     rule (s x) ≤ 0     ⟶ ⊥
 rule Prf ⊥ ⟶ Πc:Prop. Prf c              rule      0 ≤ y    ⟶ ⊤
 rule x ≤ x ⟶ ⊤                           rule (s x) ≤ (s y) ⟶ x ≤ y
```

Dependency pairs are a key concept at the core of modern automated termination provers for first-order term rewriting systems. Arts and Giesl [2] proved that a first-order rewriting relation terminates if and only if there are no infinite chains, that are sequences of dependency pairs interleaved with reductions in the arguments. We extend this notion of dependency pair to higher-order rewriting. Then we prove that, for a large class of rewriting systems $\mathcal{R}$, the combination of β and $\mathcal{R}$ is strongly normalizing on terms typable in $\lambda\Pi/\mathcal{R}$ if, there is no infinite chain.

To do so, we first construct a model of this calculus based on an adaptation of Girard's reducibility candidates [5], and prove that every typable term is strongly normalizing if every symbol of the signature is in the interpretation of its type (Adequacy lemma). We then prove that this hypothesis is verified if there is no infinite chain.

Our criterion has been implemented in SizeChangeTool. For now, it takes as input Xtc (used for the termination competition) or Dedukti files, but it could be easily adapted to a subset of other languages like Agda. As far as we know, this tool is the first one to automatically check termination in $\lambda\Pi/\mathcal{R}$, which includes both higher-order rewriting and dependent types.

Definition 1 ($\lambda\Pi/\mathcal{R}$)**.** $\lambda\Pi/\mathcal{R}$ is the PTS λP [3], enriched by a finite signature $\mathbb{F}$ and a set $\mathcal{R}$ of rules $(\Delta, f\,\vec{l} \to r)$ such that $\mathrm{FV}(r) \subseteq \mathrm{FV}(l)$ and Δ is a context associating a type to every variable of $\vec{l}$. Each $f \in \mathbb{F}$ has a type Θ_f and a sort s_f. f is (partially) defined if it is the head of the left-hand side of a rule.

Let $\to = \to_\beta \cup \to_\mathcal{R}$ where $\to_\beta$ is the β-reduction of λ-calculus and $\to_\mathcal{R}$ is the smallest relation containing $\mathcal{R}$ and closed by substitution and context. The typing rules are the ones of λP too, but the conversion is enriched with $\mathcal{R}$ and function symbol introduction is similar to variable introduction.

$$(\text{conv})\ \frac{\Gamma \vdash a : A \qquad A \to^* {}^*\!\leftarrow B \qquad \Gamma \vdash B : s}{\Gamma \vdash a : B} \qquad (\text{fun})\ \frac{\Gamma \vdash \Theta_f : s_f}{\Gamma \vdash f : \Theta_f}$$

We assume that $\to$ is locally confluent and preserves typing. For all f, we require $\vdash \Theta_f : s_f$.

As a β step can generate an $\mathcal{R}$ step, and vice versa, we cannot expect to prove the termination of $\to_\beta \cup \to_\mathcal{R}$ from the termination of $\to_\beta$ and $\to_\mathcal{R}$. The termination of $\lambda\Pi/\mathcal{R}$ cannot be reduced to the termination of the simply-typed λ-calculus either because of type-level rewriting rules.

So we build a model of our calculus by interpreting types into sets of terminating terms, adapting Girard's candidates [5]. To do so, we assume given a well-founded order on symbols of sort $\Box$. This interpretation is such that if f is a function symbol of type $\Pi(\vec{x} : \vec{T}).U\,\vec{y}$, then $f\,\vec{x} \in [\![U\,\vec{y}]\!]$ implies $x_i \in [\![T_i]\!]$ if i is an accessible position in f (accessibility is similar to the constraint of positivity of inductive types, for rewriting; see [4] for a definition).

We then extend this definition to interpret all types, such that if $T \to U$, then $[\![T]\!] = [\![U]\!]$.

We use $\sigma \vDash \Gamma$ to denote that for all $x : T$ in Γ, $\sigma(x) \in [\![T]\!]_\sigma$.

Lemma 2 (Adequacy)**.** Assuming for all f, $f \in [\![\Theta_f]\!]$, if $\Gamma \vdash t : T$ and $\sigma \models \Gamma$, then $t\sigma \in [\![T]\!]_\sigma$.

The hypothesis "for all f, $f \in [\![\Theta_f]\!]$" can be reduced to the absence of infinite chains, as shown by Arts and Giesl for first-order rewriting [2].

Definition 3 (Dependency pairs)**.** Let $f\,\vec{l} > g\,\vec{m}$ iff there is a rule $f\,\vec{l} \to r \in \mathcal{R}$, g is (partially) defined and $g\,\vec{m}$ is a maximally applied subterm of r.

$f\,t_1 \dots t_p \mathrel{\tilde{>}} g\,u_1 \dots u_q$ iff there are a dependency pair $f\,l_1 \dots l_i > g\,m_1 \dots m_j$ with $i \le p$ and $j \le q$ and a substitution σ such that, for all $k \le i$, $t_k \to^* l_k\sigma$ and, for all $k \le j$, $m_k\sigma = u_k$.

We consider a pre-order $\succeq$ on $\mathbb{F}$ compatible with typing and rewriting (*ie.* if $g \in \Theta_f$ or $r \rhd g$ for $f\,\vec{l} \to r \in \mathcal{R}$, then $f \succeq g$). $\mathcal{R}$ is well-structured and accessible if for every rule $(\Delta, f\,\vec{l} \to r)$, r is typable using only symbols smaller or equal to f and for every substitution σ, if $\Theta_f = \Pi\vec{x} : \vec{T}.U$, then $[\vec{x} \mapsto \vec{l}]\sigma \models \vec{x} : \vec{T}$ implies $\sigma \models \Delta$.

Theorem 4. The relation $\to = \to_\beta \cup \to_\mathcal{R}$ terminates on terms typable in $\lambda\Pi/\mathcal{R}$ if $\to$ is locally confluent and preserves typing, $\mathcal{R}$ is well-structured and accessible, and $\tilde{>}$ terminates.

Following [4], accessibility can be checked, by imposing that every variable occurring in the right-hand side of a rule is accessible in the left-hand side. Following [7], to prove that $\tilde{>}$ terminates, we can use Lee, Jones and Ben-Amram's size-change termination criterion [6].

Thus, we obtain a modular criterion extending Arts and Giesl's theorem that a rewriting relation terminates if there are no infinite chains [2] from first-order rewriting to dependently-typed higher-order rewriting.

This result also extends Wahlstedt's work [7] from weak to strong normalisation. Like Wahlstedt's work, Agda's termination checker [1] is designed for rules defined by constructors pattern matching, enforcing the rewriting system to be orthogonal and every definition to be total. Our criterion requires a much weaker condition: local confluence.

References

[1] A. Abel. foetus – Termination Checker for Simple Functional Programs. 1998

[2] T. Arts, J. Giesl. Termination of term rewriting using dependency pairs. *TCS* 236:133–178, 2000.

[3] H. Barendregt. Lambda calculi with types. In *Handbook of logic in computer science. Volume 2. Background: computational structures*, p. 117–309. Oxford University Press, 1992.

[4] F. Blanqui. Definitions by rewriting in the calculus of constructions. *MSCS* 15(1):37–92, 2005.

[5] J.-Y. Girard, Y. Lafont, P. Taylor. *Proofs and types*. Cambridge University Press, 1988.

[6] C. S. Lee, N. Jones, A. Ben-Amram. The size-change principle for program termination. POPL'01.

[7] D. Wahlstedt. *Dependent type theory with first-order parameterized data types and well-founded recursion*. PhD thesis, Chalmers University of Technology, 2007.