A Formalised First-Order Confluence Proof for the λ-Calculus using One-Sorted Variable Names

(*Barendregt was right after all ... almost*)

René Vestergaard[1] and James Brotherston[2]

[1] CNRS-IML, Marseille, France, vester@iml.univ-mrs.fr *
[2] University of Edinburgh, Scotland, jjb@dcs.ed.ac.uk **

Abstract. We present the titular proof development which has been implemented in Isabelle/HOL. As a first, the proof is conducted exclusively by the primitive induction principles of the standard syntax and the considered reduction relations: the naive way, so to speak. Curiously, the Barendregt Variable Convention takes on a central technical role in the proof. We also show (i) that our presentation coincides with Curry's and Hindley's when terms are considered equal up-to α and (ii) that the confluence properties of all considered calculi are equivalent.

1 Introduction

The λ-calculus is a higher-order language: terms can be abstracted over terms. It is intended to formalise the concept of a *function*. The terms of the λ-calculus are typically generated inductively thus: $\Lambda^{\mathrm{var}} ::= x \mid \Lambda^{\mathrm{var}}\Lambda^{\mathrm{var}} \mid \lambda x.\Lambda^{\mathrm{var}}$

A λ-term, $e \in \Lambda^{\mathrm{var}}$, is hence finite and is either a variable, an application of one term to another, or the functional abstraction (aka binding) of a variable over a term, respectively. On top of the terms, we define *reduction relations*, as we shall see shortly. Intuitively, we will also want to consider terms that only differ in the particular names used to express abstraction to be equal. However, this is a slightly tricky construction as far as the algebra of the syntax goes and we will only undertake it after mature consideration.

It is common, informal practice to take the variables to belong to a single infinite set of names, $\mathcal{VN}$, with a decidable equality relation, $=$, and that is indeed what we will do. Recent research [8, 17] has shown that there can be formalist advantages to employing a certain amount of ingenuity on the issue of variable names. Still, we make a point of following the naive approach. In fact, the main contribution of this paper is to show that it is not only possible but also feasible and even instructive to use this, the naive set-up, for formal purposes. This is relevant both from a foundational and a practical perspective. The latter more-so as we, as a first, give a rational reconstruction of the widely used and very helpful Barendregt Variable Convention (BVC) [1].

* Supported under EU TMR grant # ERBFMRXCT-980170: LINEAR. Work done in part while visiting LFCS, University of Edinburgh from Heriot-Watt University.
** Supported by a grant from LFCS, University of Edinburgh.

We stress that Λ^{var} is first-order abstract syntax (FOAS) and therefore comes equipped with a primitive (first-order) principle of *structural induction* [2]:

$$\frac{\forall x.P(x) \quad \forall e_1, e_2.P(e_1) \wedge P(e_2) \to P(e_1 e_2) \quad \forall x, e.P(e) \to P(\lambda x.e)}{\forall e.P(e)}$$

Similarly, the syntax also comes equipped with a primitive recursion principle so we can define auxiliary notions (e.g., free variables) by case-splitting.

The Issues In the set-up of FOAS defined over one-sorted variable names ($\mathrm{FOAS}_{\mathcal{VN}}$), name-overlaps seem inevitable when computing. Traditionally, one therefore renames offending binders when appropriate. This has a two-fold negative impact: (i) the notion 'sub-term of' on which structural induction depends is typically broken,[1] and (ii) as a term can reduce in different directions, the resulting name for a given abstraction cannot be pre-determined. Consider, e.g., the following example taken from [11] — for precise definitions see Section 1.2:

$$(\lambda x.(\lambda y.\lambda x.xy)x)y \begin{array}{l} \xrightarrow{\beta^{\mathrm{C}}} (\lambda y.\lambda x.xy)y \xrightarrow{\beta^{\mathrm{C}}} \lambda x.xy \\ \xrightarrow{\beta^{\mathrm{C}}} (\lambda x.\lambda z.zx)y \xrightarrow{\beta^{\mathrm{C}}} \lambda z.zy \end{array}$$

Equational reasoning about $\mathrm{FOAS}_{\mathcal{VN}}$ can thus seemingly only be conducted up-to post-fixed "name-unification". Aside from any technical problems this might pose, the formal properties we establish require some interpretation.

The basic problems with $\mathrm{FOAS}_{\mathcal{VN}}$ has directly resulted in the inception of syntax formalisms (several of them recent) which overcome the issues by native means [4–8, 12]. In general, they mark a conceptual and formal departure from the naive qualities of $\mathrm{FOAS}_{\mathcal{VN}}$. This is in part unfortunate because $\mathrm{FOAS}_{\mathcal{VN}}$ is the de facto standard in programming language theory where, as a result of the problems, it is customary to reason while "assuming the BVC" [1]:[2]

> "**2.1.12.** Terms that are α-[equivalent] are identified."
> "**2.1.13.** If $M_1, \ldots, M_n$ occur in a certain mathematical context, [their] bound variables are chosen to be different from the free variables."
> "**2.1.14.** Using 2.1.12/13 one can work with λ-terms the naive way."

Our Contribution We

- show that it is possible and feasible to conduct formal equational proofs about higher-order languages by simple, first-order means
- show that this can be done over $\mathrm{FOAS}_{\mathcal{VN}}$, as done by hand
- formally justify informal practices; in particular, the BVC [1, 24]
- contribute to a much needed proof theoretical analysis of binding [8, 9]
- introduce a quasi-complete range of positive and negative results about the preservation and reflection of confluence under a large class of mappings

[1] Thanks to Regnier for observing that this need not happen with parallel substitution.
[2] We make reference to Barendregt because it is common practice to do so. Many other people have imposed hygiene conditions on variables.

1.1 Terminology and Conventions

We say that a term *reduces* to another if they are related by a reduction relation and we denote it by an infix arrow. The sub-term a reduction step acts upon is called the *redex* and it is said to be *contracted*. A reduction relation for which a redex remains so when occurring in any sub-term position is said to be *contextually closed*. We will distinguish *raw* and *real* calculi: inductive structures vs. the former factored by an equivalence. We use dashed respectively full-lined relational symbols for them. The first 5 of the following notions can be given by proper inductive constructions.

- The converse of a relation, $\rightarrow$, is written $(\rightarrow)^{-1}$.
- Composition is: $a \rightarrow_1 ; \rightarrow_2 c \Leftrightarrow^{\text{def}} \exists b \,.\, a \rightarrow_1 b \ \wedge \ b \rightarrow_2 c$.
- Given two reduction relations $\rightarrow_1$ and $\rightarrow_2$, we have: $\rightarrow_{1\cup 2} =^{\text{def}} \rightarrow_1 \cup \rightarrow_2$.
- Transitive, reflexive closures: $(\rightarrow)^\star \ =^{\text{def}} \twoheadrightarrow =^{\text{“def”}} = \cup(\rightarrow ; \twoheadrightarrow)$.
- Transitive, reflexive, and symmetric closures: $=_A =^{\text{def}} \ (\rightarrow_A \cup (\rightarrow_A)^{-1})^\star$.
- A relation which is functional will be written with a based arrow: $\mapsto$.
- A term reducing to two terms is called a *divergence*.
- Two diverging reduction steps, as defined above, are said to be *co-initial*.
- Two reduction steps that share their end-term are said to be *co-final*.
- A divergence is *resolvable* if there exist connecting co-final reduction steps.
- A relation has the *diamond property*, $\diamond$, if any divergence can be resolved.
- A relation, $\rightarrow$, is *confluent*, Confl, if $\diamond(\twoheadrightarrow)$.
- *Weak confluence* is transitive, reflexive resolution of any divergence.
- An abstract rewrite systems, *ARS*, is a relation on a set: $\rightarrow \subseteq A \times A$.
- *Residuals* are the descendants of terms under reduction.

1.2 Classic Presentations of the λ-Calculus

We will here review Curry's seminal formalist presentation of the λ-calculus [3]. We will also review Hindley [11] as, to the best of our knowledge, he is the first to give serious consideration to the problems with names in equational proofs. The process of term substitution epitomises the issues. The two Λ^{var}-terms: $\lambda x.y$ and $\lambda z.y$, e.g., have the same intuitive meaning. If we intend to substitute, say, the term x in for y, simple syntactic replacement would result in the intuitively different terms: $\lambda x.x$ and $\lambda z.x$. Some subtlety is therefore required.

Curry's Presentation Curry [3] essentially defines the terms of the λ-calculus to be Λ^{var} with the proviso that variable names are ordered linearly. He defines substitution as follows — for *free variables*, $\text{FV}(-)$, see Section 3, Figure 1:

$$y\langle x := e\rangle = \begin{cases} e & \text{if } x = y \\ y & \text{otherwise} \end{cases}$$

$$(e_1 e_2)\langle x := e\rangle = e_1\langle x := e\rangle e_2\langle x := e\rangle$$

$$(\lambda y.e')\langle x := e\rangle = \begin{cases} \lambda y.e' & \text{if } x = y \\ \lambda y.e'\langle x := e\rangle & \text{if } x \neq y \wedge (y \notin \text{FV}(e) \vee x \notin \text{FV}(e')) \\ \lambda z.e'\langle y := z\rangle\langle x := e\rangle & \text{o/w; } \textit{first } z \notin \{x\} \cup \text{FV}(e) \cup \text{FV}(e') \end{cases}$$

Curry is seminal in giving a precise definition of substitution which takes into account that scoping is static. He then defines the following reduction relations for λ which are closed contextually:

- $\lambda y.e\langle x := y\rangle \dashrightarrow_{\alpha^{\mathrm{C}}} \lambda x.e$, if $y \notin \mathrm{FV}(e)$
- $(\lambda x.e)e' \dashrightarrow_{\beta^{\mathrm{C}}} e\langle x := e'\rangle$

Unfortunately, following on from here, Curry makes no further mentioning of α in the proofs of the equational properties of the λ-calculus. Instead, all proofs are seemingly conducted implicitly on α-equivalence classes although these are not formally introduced.

Hindley's Presentation This situation, amongst others, was rectified by Hindley [11]. In order to address α-equivalence classes explicitly, Hindley introduced a restricted α-relation which we call α^{H}. The relation is given as the contextual closure of:

- $\lambda x.e \dashrightarrow_{\alpha^{\mathrm{H}}} \lambda y.e\langle x := y\rangle$, if $x \neq y$, $y \notin \mathrm{FV}(e) \cup \mathrm{BV}(e)$, and $x \notin \mathrm{BV}(e)$

The α^{H}-relation has the nice property that the renaming clause of $-\langle - := -\rangle$ is not invoked, cf. Lemma 9. Furthermore, a number of Hindley's results conspire to establish the following property:

Lemma 1 (From Lemma 4.7, Lemma 4.8, Corollary 4.8 [11])

$$==_{\alpha^{\mathrm{C}}} \;=\; \dashrightarrow\!\!\!\!\!\rightarrow_{\alpha^{\mathrm{C}}} \;=\; \dashrightarrow\!\!\!\!\!\rightarrow_{\alpha^{\mathrm{H}}} \;=\; ==_{\alpha^{\mathrm{H}}}$$

Notation 2 *To have an axiomatisation-independent name for α-equivalence on Λ^{var}, we will also refer to the relation of the above lemma as $\aleph$ (read: aleph).*

With this result in place, Hindley undertakes a formal study of α-equivalence classes which leads to the definition of a further β-relation, this time on α^{H}-equivalence classes:

$$\lfloor e \rfloor_{\mathrm{H}} =^{\mathrm{def}} \{e' \mid e ==_{\alpha^{\mathrm{H}}} e'\}$$
$$\lfloor e_1 \rfloor_{\mathrm{H}} \rightarrow_{\beta^{\mathrm{H}}} \lfloor e_2 \rfloor_{\mathrm{H}} =^{\mathrm{def}} \exists e_1' \in \lfloor e_1 \rfloor_{\mathrm{H}}, e_2' \in \lfloor e_2 \rfloor_{\mathrm{H}}.e_1' \dashrightarrow_{\beta^{\mathrm{C}}} e_2'$$

It is this relation which Hindley proves confluent albeit with no formal considerations concerning the invoked proof principles. This puts Hindley's treatment of the λ-calculus firmly apart from the present article. Interestingly, Hindley also points out that the obtained (real) confluence result implies confluence of the combined α^{C}- and β^{C}-relation. We are able to formally substantiate this remark of Hindley, cf. Theorem 16.

1.3 Related Work

There has recently been a substantial amount of work on proof principles for syntax that seemingly are more advanced than the first-order principles we use [4–8, 10, 12]. These lines of work, in particular the continuations of [6, 8, 22], are all very interesting but orthogonal to the work we present here. We suggest that a study of the proof-theoretical strength of these different proof principles might be informative and we leave it as potential future work.

There are a number of formalisations of β-confluence [13, 17, 18, 22, 23]. Apart from [17] which uses two-sorted variable names to distinguish bound and free variables, they all use either higher-order abstract syntax or de Bruijn indexing.

We know of no formal proof developments, be it for equational properties or otherwise, that are based on $\mathrm{FOAS}_{\mathcal{VN}}$. That said, Schroer [21] does undertake a hand-proof of confluence of the λ-calculus which explicitly pays attention to variable names but with its 700+ pages, it is perhaps not as approachable as could be desired. Besides, no particular attention is paid to the employed proof principles and no formalisation is undertaken.

1.4 Acknowledgements

The first author wishes to thank Olivier Danvy, Jean-Yves Girard, Stefan Kahrs, Don Sannella, Randy Pollack, and in particular Joe Wells for fruitful discussions. The second author wishes to thank James Margetson, Larry Paulson, and Markus Wenzel for help and advice on using Isabelle/HOL. Finally, both authors wish to thank LFCS and the anonymous referees.

1.5 A Word on our Proofs

The Isabelle/HOL proof development underpinning the present article was undertaken mainly by the second author in the space of roughly 9 weeks. It is available from the first author's homepage. At the time of writing, the confluence properties for our λ^{var}-calculus (Section 3) and the λ-calculus proper have been established. The Isabelle proof development closely follows the presentation we give here. There are one or two differences which are exclusively related to the use of alternative but equivalent induction principles in certain situations.

We started from scratch and learned theorem proving and Isabelle as we went along. Our proofs are mainly brute-force in that Isabelle apparently had problems overcoming the factorial blow-up in search space arising from the heavily conditioned proof goals for our conditional rewrite rules. Presently, the size of our proof scripts is in the order of 4000 lines of code.

The second author's Honours project will contain more detailed information about the proof development itself and will focus in part on the automation issue. The first author's thesis will focus more generally on first-order equational reasoning about higher-order languages.

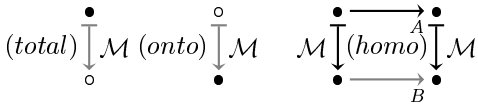
2 Abstract Proof Techniques for Confluence

We now present the (new and well-known) abstract rewriting methods we use.

2.1 Preservation and Reflection of Confluence

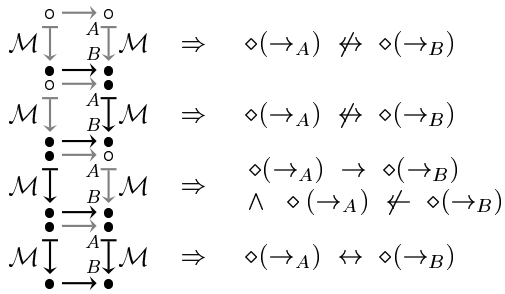
Surprisingly, the results in this section seems to be new. Although they are very basic and related to the areas of rewriting modulo and refinement theory, we have not found any comprehensive overlaps.[3] In any event, the presentation is novel and instructive for the present purposes. Before proceeding, we refer the reader to Appendix A for an explanation of our diagram notation.

Definition 3 (Ground ARS Morphism) *Assume two ARS:* $\rightarrow_A \subseteq A \times A$ *and* $\rightarrow_B \subseteq B \times B$*. A mapping,* $\mathcal{M} : A \longrightarrow B$*, will be said to be a* ground ARS morphism[4] *from* $\rightarrow_A$ *to* $\rightarrow_B$ *if it is total and onto on points and a homomorphism from* $\rightarrow_A$ *to* $\rightarrow_B$*:*



An example of a ground ARS morphism is the function that sends an object to its equivalence class relative to any equivalence relation (such as, α- or AC-equivalence): what one would call a "structural collapse". Notice that a ground ARS morphism prescribes surjectivity on objects but not on relations (and, as such, should not be called a "structural collapse" in itself). Instead, the following theorem analyses the various "degrees of relational surjectivity" relative to the confluence property.

Theorem 4 *Given a ground ARS morphism,* $\mathcal{M}$*, from* $\rightarrow_A$ *to* $\rightarrow_B$*, we have:*[5]



Proof The positive results are straightforward to establish. The reflexive(!) versions of the following ARS provide counter-examples for all the negative results, left-to-right and right-to-left, respectively. Reflexivity is required to establish the $\diamond$ property in the first place.

[3] A special case of Theorem 4, 4 is reported in [14] and we contradict a result in [19].
[4] The name is inspired from [20].
[5] In the theorem, the notation $\not\rightarrow$ ($\not\Rightarrow$) means existence of counter-examples.

□

The asymmetry between cases 2 and 3 is due to the functionality of $\mathcal{M}$.

Implications In order to preserve confluence under a "structural collapse" (i.e., a ground ARS morphism plus a premise from Theorem 4), we see from Theorem 4, cases 1 and 2 that it is insufficient to simply prove a raw diamond property which admit an initiality condition on well-formedness of raw terms. Observe that this is exactly what happens in the wider programming language community when using the BVC.

2.2 Resolving the Abstract Proof Burden of Confluence

We now sketch the abstract part of the Tait/Martin-Löf proof method for confluence as formalised by Nipkow [18] plus what we call Takahashi's Trick [24].

A Formalisation of the Tait/Martin-Löf Method The Tait/Martin-Löf proof method uses a parallel relation that can contract any number of pre-existing redexes in one step, cf. Figure 4. The crucial step in applying the method is the following property of ARS.

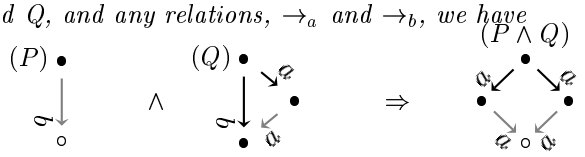
Lemma 5 $(\exists \to_2 \,.\, \to_1 \subseteq \to_2 \subseteq \twoheadrightarrow_1 \;\wedge\; \diamond(\to_2)) \;\Rightarrow\; \mathrm{Confl}(\to_1)$

Proof A formalisation is provided in [18] and is re-used here. □

The point is that, since a parallel relation, $\to_2$ above, can contract an arbitrary number of redexes in parallel, only one reduction step is required to contract the unbounded copies of a particular redex that could have been created through duplication by a preceding reduction.

Takahashi's Trick In order to prove the diamond property of a parallel β-relation, Takahashi [24] introduced the trick of using an inductively defined *complete development* relation, cf. Figure 5, rather than proceed by direct means (i.e., an involved case-splitting on the relative locations of redexes). Instead of resolving a parallel divergence "minimally" (i.e., by a brute-force case-splitting), Takahashi's idea is to go for "maximal" resolution: the term that has all pre-existing redexes contracted in one step is co-final for any parallel divergence. Abstractly, the following ARS result underpins Takahashi's idea up-to the guarding predicates which we have introduced.

Lemma 6 (Takahashi's Diamond Diagonalisation (Guarded)) *For any predicates, P and Q, and any relations, $\to_a$ and $\to_b$, we have*



$$
\begin{aligned}
y[x := e] &= \begin{cases} e \text{ if } x = y \\ y \text{ otherwise} \end{cases} \\
(e_1 e_2)[x := e] &= e_1[x := e] e_2[x := e] \\
(\lambda y.e')[x := e] &= \begin{cases} \lambda y.e'[x := e] \text{ if } x \neq y \wedge y \notin \mathrm{FV}(e) \\ \lambda y.e' \qquad\qquad \text{otherwise} \end{cases}
\end{aligned}
$$

$$
\begin{array}{ll}
\mathrm{FV}(y) = \{y\} & \mathrm{Capt}_x(y) = \emptyset \\
\mathrm{FV}(e_1 e_2) = \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) & \mathrm{Capt}_x(e_1 e_2) = \mathrm{Capt}_x(e_1) \cup \mathrm{Capt}_x(e_2) \\
\mathrm{FV}(\lambda y.e) = \mathrm{FV}(e) \setminus \{y\} & \mathrm{Capt}_x(\lambda y.e) = \begin{cases} \{y\} \cup \mathrm{Capt}_x(e) \text{ if } x \neq y \wedge x \in \mathrm{FV}(e) \\ \emptyset \qquad\qquad\qquad\quad \text{otherwise} \end{cases}
\end{array}
$$

Fig. 1. Total but partially correct substitution, $-[- := -]$, free variables, $\mathrm{FV}(-)$, and variables capturing free occurrences of x, $\mathrm{Capt}_x(-)$, for Λ^{var}.

$$
\frac{y \notin \mathrm{Capt}_x(e) \cup \mathrm{FV}(e)}{\lambda x.e \overset{y}{\dashrightarrow}_{i\alpha} \lambda y.e[x := y]}\,(\alpha) \qquad \frac{\mathrm{FV}(e_2) \cap \mathrm{Capt}_x(e_1) = \emptyset}{(\lambda x.e_1) e_2 \dashrightarrow_{\beta} e_1[x := e_2]}\,(\beta)
$$

Fig. 2. Raw β- and indexed α-contraction — reduction is given by full contextual closure. By the premises no invoked substitution will result in free-variable capture.

Proof Straightforward. □

The second premise is often called the triangle property when $\rightarrow_b$ is functional.

3 The λ^{var}-Calculus

We will now formally define the λ^{var}-calculus and go on to show that its "structural collapse" under α is the λ-calculus proper as defined in Section 1.2.

Definition 7 (The λ^{var}-Calculus) *The terms of the λ^{var}-calculus are Λ^{var}, given on page 1.* Substitution, free variables *and* capturing variables *of raw terms are defined in Figure 1. The β- and indexed α-rewriting relations of λ^{var}: $\dashrightarrow_{\beta}$ and $\overset{-}{\dashrightarrow}_{i\alpha}$ are given inductively by contextual closure from Figure 2. Plain α-rewriting is given as: $e_1 \dashrightarrow_{\alpha} e_2 \Leftrightarrow^{\mathrm{def}} \exists z.e_1 \overset{z}{\dashrightarrow}_{i\alpha} e_2$*

The indexed α-rewriting relation will be used to conduct the ensuing proofs but is, as such, not needed for defining the λ^{var}-calculus. We stress that the (inductively defined) reduction relations also come equipped with first-order induction principles. We will typically refer to uses of these as *rule induction*. The main novelty in the above definition is the side-conditions on the contraction rules that makes binder-renaming unnecessary. The construct $\mathrm{Capt}_x(e)$ returns all the binding variables in e that have a free occurrence of x (relative to e) in their scope. It coincides with the all but forgotten notion of *not free for*. Substitution has been defined the way it has purely to enable us to prove certain "renaming sanity" properties for it, which we however will not present here.

$\mathrm{BV}(x) = \emptyset$	$\mathrm{UB}(x) = \mathrm{True}$
$\mathrm{BV}(e_1 e_2) = \mathrm{BV}(e_1) \cup \mathrm{BV}(e_2)$	$\mathrm{UB}(e_1 e_2) = \mathrm{UB}(e_1) \wedge \mathrm{UB}(e_2) \wedge \mathrm{BV}(e_1) \cap \mathrm{BV}(e_2) = \emptyset$
$\mathrm{BV}(\lambda x.e) = \mathrm{BV}(e) \cup \{x\}$	$\mathrm{UB}(\lambda x.e) = \mathrm{UB}(e) \wedge x \notin \mathrm{BV}(e)$

Fig. 3. The bound variables and the uniquely bound predicate for the terms of Λ^{var}.

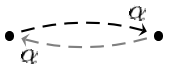
Proposition 8 $\mathrm{FV}(e_2) \cap \mathrm{Capt}_x(e_1) = \emptyset \;\Rightarrow\; e_1[x := e_2] = e_1\langle x := e_2\rangle$

Proof By structural induction in e_1. The only non-trivial case is $e_1 \equiv \lambda y.e_1'$ which is handled by a tedious case-splitting on y. The main case is $y \neq x$ and $y \in \mathrm{FV}(e_2)$. Here, the premise of the proposition means that $y \notin \mathrm{Capt}_x(\lambda y.e')$ which immediately implies that $x \notin \mathrm{FV}(e')$ by $y \neq x$. We hence avoid $-\langle - := -\rangle$ performing a binder renaming. □

Lemma 9 $\dashrightarrow_{\alpha^{\mathrm{H}}} \subseteq \dashrightarrow_{\alpha} \subseteq (\dashrightarrow_{\alpha^{\mathrm{C}}})^{-1}$

Proof The first inclusion follows as the side-condition on $\dashrightarrow_{\alpha^{\mathrm{H}}}$ is subsumed by the side-condition on $\dashrightarrow_{\alpha}$. Any invoked substitutions thus coincide by Proposition 8 whose premise is established by the latter's side-condition. The reasoning for the second inclusion is analogous. □

Lemma 10 ($\dashrightarrow_{\alpha}$-Symmetry)



Lemma 11 $\aleph = \dashrightarrow\!\!\!\!\twoheadrightarrow_{\alpha} = ==_{\alpha}$

Proof From Lemmas 1 and 9 and Lemma 10, respectively. □

Lemma 12 $\dashrightarrow_{\beta} \subseteq \dashrightarrow_{\beta^{\mathrm{C}}} \subseteq \dashrightarrow\!\!\!\!\twoheadrightarrow_{\alpha}; \dashrightarrow_{\beta}$

Proof The first inclusion follows from Proposition 8. The second follows by observing that all the renamings required to perform the β^{C}-induced substitution preserve α^{C}-equivalence, i.e., $\aleph$-equivalence. By Lemma 11, they can thus be expressed by $\dashrightarrow\!\!\!\!\twoheadrightarrow_{\alpha}$. It suffices to observe that no renaming is performed following the "passing" of the substitution invoked by the β-rule. □

λ^{var} α-Collapses to the Real λ-Calculus With these fundamental results in place, we have ensured the intuitive soundness of the following definition — which mimics Hindley's construction.

Definition 13 (The Real λ-Calculus)

- $\Lambda = \Lambda^{\mathrm{var}} / ==_{\alpha}$
- $\lfloor - \rfloor : \Lambda^{\mathrm{var}} \longrightarrow \Lambda$
  $e \quad \mapsto \{e' \mid e ==_{\alpha} e'\}$
- $\lfloor e \rfloor \to_{\beta} \lfloor e' \rfloor \Leftrightarrow^{def} e ==_{\alpha}; \dashrightarrow_{\beta}; ==_{\alpha} e'$

$$\frac{}{x \;\text{-}\!\!\twoheadrightarrow_\beta\; x} \qquad \frac{e \;\text{-}\!\!\twoheadrightarrow_\beta\; e'}{\lambda x.e \;\text{-}\!\!\twoheadrightarrow_\beta\; \lambda x.e'} \qquad \frac{e_1 \;\text{-}\!\!\twoheadrightarrow_\beta\; e_1' \qquad e_2 \;\text{-}\!\!\twoheadrightarrow_\beta\; e_2'}{e_1 e_2 \;\text{-}\!\!\twoheadrightarrow_\beta\; e_1' e_2'}$$

$$\frac{e_1 \;\text{-}\!\!\twoheadrightarrow_\beta\; e_1' \qquad e_2 \;\text{-}\!\!\twoheadrightarrow_\beta\; e_2' \qquad \mathrm{FV}(e_2') \cap \mathrm{Capt}_x(e_1') = \emptyset}{(\lambda x.e_1) e_2 \;\text{-}\!\!\twoheadrightarrow_\beta\; e_1'[x := e_2']}$$

Fig. 4. The *parallel* β-relation: arbitrary, pre-existing β-redexes contracted in parallel.

Following on from the definition, we see that we have:

Proposition 14 $\lfloor e \rfloor \twoheadrightarrow_\beta \lfloor e' \rfloor \;\Leftrightarrow\; e\ (==_\alpha; \text{--}\!\!\rightarrow_\beta; ==_\alpha)^\star\ e' \;\vee\; e ==_\alpha e'$

Proof The left-most disjunct is the straightforward transitive version of our definition of real β. The right-most disjunct comes from the reflexive case, again by definition. □

We thus arrive at the following, rather appeasing, result.

Lemma 15 $\lfloor e \rfloor \twoheadrightarrow_\beta \lfloor e' \rfloor \Leftrightarrow e \text{--}\!\!\twoheadrightarrow_{\alpha \cup \beta} e' \Leftrightarrow e \text{--}\!\!\twoheadrightarrow_{\alpha^{\mathrm{C}} \cup \beta^{\mathrm{C}}} e' \Leftrightarrow \lfloor e \rfloor_{\mathrm{H}} \twoheadrightarrow_{\beta^{\mathrm{H}}} \lfloor e' \rfloor_{\mathrm{H}}$

Proof From Lemma 10, it is trivial to see that $(==_\alpha; \text{--}\!\!\rightarrow_\beta; ==_\alpha)^\star \cup ==_\alpha = \text{--}\!\!\twoheadrightarrow_{\alpha \cup \beta}$ and the first biimplication is established by Proposition 14. The second biimplication follows by Lemmas 9, 11, and 12. The last biimplication follows in an analogous manner. □

Equivalence of the Raw and the Real Calculi The technical reason for calling the above result "appeasing" is that it allows us to prove the equational equivalence results for the raw and the real calculi we have made reference to. We consider the second result to be of particular interest.

Theorem 16

- $(\Lambda / =_\beta) \;=\; (\Lambda^{\mathrm{var}} / ==_{\alpha \cup \beta}) \;=\; (\Lambda^{\mathrm{var}} / ==_{\alpha^{\mathrm{C}} \cup \beta^{\mathrm{C}}}) \;=\; ((\Lambda^{\mathrm{var}} / ==_{\alpha^{\mathrm{H}}}) / =_{\beta^{\mathrm{H}}})$
- $\mathrm{Confl}(\rightarrow_\beta) \;\leftrightarrow\; \mathrm{Confl}(\text{--}\!\!\rightarrow_{\alpha \cup \beta}) \;\leftrightarrow\; \mathrm{Confl}(\text{--}\!\!\rightarrow_{\alpha^{\mathrm{C}} \cup \beta^{\mathrm{C}}}) \;\leftrightarrow\; \mathrm{Confl}(\rightarrow_{\beta^{\mathrm{H}}})$

Proof The first result is immediate following Lemma 15. As for the second result, the definitional totality and surjectivity of $\lfloor - \rfloor$ and $\lfloor - \rfloor_{\mathrm{H}}$ combined with Lemma 15 allow us to apply Theorem 4, case 4. □

Having thus formally convinced ourselves that we are about to solve the right problem, we will now present the details of the confluence proof.

4 An Equational λ^{var}-Property and λ-Confluence

As outlined in Sections 2 and 3, it suffices to find a raw relation over Λ^{var} which enjoys the diamond property in order to prove the confluence property for the λ-calculus. Taking the lead from the Tait/Martin-Löf method, this relation needs to contain a notion of parallel β-reduction.

$$\frac{}{x \dashrightarrow_{|\beta} x} \qquad \frac{e \dashrightarrow_{|\beta} e'}{\lambda x.e \dashrightarrow_{|\beta} \lambda x.e'} \qquad \frac{e \dashrightarrow_{|\beta} e'}{xe \dashrightarrow_{|\beta} xe'} \qquad \frac{e_1 e_2 \dashrightarrow_{|\beta} e' \quad e_3 \dashrightarrow_{|\beta} e'_3}{(e_1 e_2)e_3 \dashrightarrow_{|\beta} e' e'_3}$$

$$\frac{e_1 \dashrightarrow_{|\beta} e'_1 \quad e_2 \dashrightarrow_{|\beta} e'_2 \quad \mathrm{FV}(e'_2) \cap \mathrm{Capt}_x(e'_1) = \emptyset}{(\lambda x.e_1)e_2 \dashrightarrow_{|\beta} e'_1[x := e'_2]}$$

Fig. 5. The *complete development* β-relation: attempted contraction of all redexes.

Definition 17 Parallel *β-reduction, $\dashrightarrow_{\beta}$, is defined in Figure 4.*

The parallel β-relation admits the contraction of any number (including 0) of pre-existing β-redexes starting from within as long as no variable renaming is required. To give an impression of the level of detail of the formalisation, we can mention that the property which we need the most in the proof development is the following variable monotonicity result about the parallel β-relation:

Proposition 18 $e \dashrightarrow_{\beta} e' \Rightarrow \mathrm{FV}(e') \subseteq \mathrm{FV}(e) \wedge \mathrm{BV}(e') \subseteq \mathrm{BV}(e)$

In order to employ Takahashi's Trick, we need to ensure that any considered β-divergence can be resolved by a complete development step.

Definition 19 Complete β-development*, $\dashrightarrow_{|\beta}$, is defined in Figure 5.*

Observe, informally, that $\dashrightarrow_{|\beta}$ only is defined if all (pseudo-)redexes validate the side-condition on the β-rule. Or, more precisely, the relation is defined if it is possible to contract all (pseudo-)β-redexes starting from within — we will shortly show that this is indeed possible. For now, we merely present:

Lemma 20 $\dashrightarrow_{|\beta} \subseteq \dashrightarrow_{\beta}$

Proof Straightforward. □

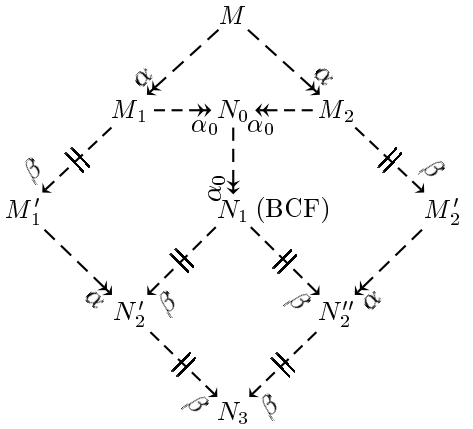
The Overall Proof Structure Having thus established the basics, we outline the proof of the diamond property of the following relation: $\dashrightarrow\!\!\!\!\rightarrow_{\alpha}; \dashrightarrow_{\beta}$, before supplying the actual details of the proof. The relation is inspired by the definitional reflection of the weak confluence property for the λ-calculus proper over the structural (α-)collapse of λ^{var}. In order to use the BVC in our proof, we first present it as a predicate on Λ^{var}, cf. Figure 3.

Definition 21 (Barendregt Conventional Form)

$$\mathrm{BCF}(e) = \mathrm{UB}(e) \wedge (\mathrm{BV}(e) \cap \mathrm{FV}(e) = \emptyset)$$

Lemma 22 $\diamond(\dashrightarrow\!\!\!\!\rightarrow_{\alpha}; \dashrightarrow_{\beta})$

Proof
For the Ms given, we can construct the Ns in the divergence resolution on the right in order. The ensuing sections will detail the individual diagrams. The $\dashrightarrow_{\alpha_0}$-relation is introduced in Definition 25 as the fresh-naming restriction of $\dashrightarrow_{\alpha}$. It serves to facilitate the commutativity with β on either side of the diagram. We note that the result means that it suffices to address all naming issues before the combinatorially more complex β-divergence which can be addressed in isolation due to BCF-initiality.



□

4.1 Substitutivity and Substitution Results

When proving a commutativity result about two relations, you typically proceed by rule induction over one of the relations. In what amounts to the non-trivial sub-cases of such a proof you therefore typically need to show that a substitution from the case-instantiated relation "distributes" over the other relation. Such results are called Substitutivity Lemmas. The non-trivial sub-cases of Substitutivity Lemmas, in turn, are called Substitution Lemmas. They establish commutativity of the substitutions from both the case-instantiations. Substitutivity and Substitution Lemmas are non-trivial to prove formally. For our present purposes we will merely display one of each to give an indication of the style. The key to understanding the following lemmas is the fact that $\mathrm{Capt}_x(e_1) \cap \mathrm{FV}(e_2) = \emptyset$ is the weakest predicate ensuring the correctness of substituting e_2 into e_1 for x.

Lemma 23 (Substitution)

$$\begin{array}{l} y \notin \mathrm{FV}(e_2) \wedge x \neq y \wedge (\mathrm{Capt}_x(e_3) \cap \mathrm{FV}(e_2) = \emptyset) \wedge (\mathrm{Capt}_y(e_1) \cap \mathrm{FV}(e_3) = \emptyset) \\ \wedge (\mathrm{Capt}_x(e_1) \cap \mathrm{FV}(e_2) = \emptyset) \wedge (\mathrm{Capt}_x(e_1[y := e_3]) \cap \mathrm{FV}(e_2) = \emptyset) \\ \Downarrow \\ e_1[y := e_3][x := e_2] = e_1[x := e_2][y := e_3[x := e_2]] \end{array}$$

Lemma 24 (Parallel β Substitutivity)

$$\begin{array}{l} e_1 \ \text{-H}\!\!\rightarrow_\beta e_1' \wedge e_2 \ \text{-H}\!\!\rightarrow_\beta e_2' \wedge (\mathrm{Capt}_x(e_1) \cap \mathrm{FV}(e_2) = \emptyset) \wedge (\mathrm{Capt}_x(e_1') \cap \mathrm{FV}(e_2') = \emptyset) \\ \Downarrow \\ e_1[x := e_2] \ \text{-H}\!\!\rightarrow_\beta e_1'[x := e_2'] \end{array}$$

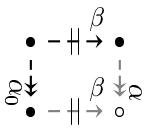
We refer the interested reader to the complete Isabelle/HOL proof development at the homepage of the first author for full details.

4.2 Weak α- and β-Commutativity

In this section we prove the lemma that is needed on either side of the diagram in the proof of Lemma 22. In trying to prove a general α and β commutativity result, we are immediately stopped by the following naming issue: for virtually all Λ^{var}-terms, there exist α-reductions that can invalidate a previously validated side-condition on a β-redex. Fortunately, we can see that the commutativity result we need concerns arbitrary β-reductions but only α-reductions that suffice to prove Lemma 22. We therefore define a restricted, fresh-naming α-relation. The definition can also be given inductively.

Definition 25 $e \dashrightarrow_{\alpha_0} e' \Leftrightarrow^{\mathrm{def}} \exists z.e \overset{z}{\dashrightarrow}_{i\alpha} e' \wedge z \notin \mathrm{FV}(e) \cup \mathrm{BV}(e)$

Lemma 26



Proof By rule induction in $\dashrightarrow\!\!\!\!\!\rightarrow_{\alpha_0}$ with the induction step going through painlessly by freshness of the relevant z. □

4.3 The Diamond Property of Parallel β up-to BVC-Initiality

We will now establish the lower part of the diagram in the proof of Lemma 22. It is proved using Takahashi's Trick, cf. Lemma 6. Initially, we thus need to establish the conditional existence of a non-renaming complete β-development.

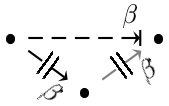
Lemma 27 (BCF) $\bullet \overset{\beta}{\dashrightarrow} \circ$

Proof By structural induction using Proposition 18 and Lemma 20. □

We stress that the proof is straightforward using the referenced variable monotonicity results as $\dashrightarrow_\beta$ is inductively defined to contract from within. No complicated considerations concerning residuals are required. However, BCF-initiality is crucial for the property. The terms $(\lambda x.\lambda y.x)y$ and $\lambda y.(\lambda x.\lambda y.x)y$ fail to enjoy free/bound variable disjointness and unique binding, respectively, and neither completely develop. BCF-initiality is thus sufficient for the existence of a complete development but only necessary in a weak sense: breaking either conjunct of the BCF-predicate can prevent renaming-free complete development. Still, some non-BCFs completely develop, e.g., $(\lambda x.x)x$ and $\lambda x.(\lambda x.x)x$.

The second of the two required results for the application of Lemma 6 must establish that any parallel β-step always can "catch up" with a completely developing β-step by a parallel β-step, *with no renaming involved*.

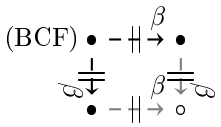
Lemma 28



Proof By rule induction in $\dashrightarrow\!\!\!\!\!\rightarrow_{\beta}$ using Lemma 24. □

It is interesting that the above property requires no initiality conditions, like the BCF-predicate, to be provable — except, that is, from well-definedness of $\dashrightarrow\!\!\!\!\!\rightarrow_{\beta}$. This is mainly due to our use of the weakest possible side-condition on β-contraction to make β renaming free (i.e., $\mathrm{FV}(-) \cap \mathrm{Capt}_{\_}(-) = \emptyset$). Had we instead required that the free variables of the argument were disjoint from the full set of bound variables in the body of the applied function (i.e., $\mathrm{FV}(-) \cap \mathrm{BV}(-) = \emptyset$), the property would not have been true. A counter-example is $(\lambda y.(\lambda x.y)z)\lambda z.z$. It takes advantage of complete developments contracting from within. Contracting the outermost redex first (e.g., by a parallel step) blocks the contraction of the residual of the innermost redex when the stronger side-condition is imposed: $(\lambda x.\lambda z.z)z$. No variable conflict is created between two residuals of the same term due to Hyland's Disjointness Property [15].[6]

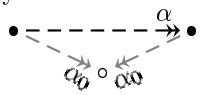
Lemma 29

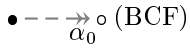


Proof From Lemmas 27 and 28 by using Takahashi's Trick, Lemma 6. □

4.4 Fresh-Naming α-Confluence with BVC-Finality

The last result we need for the proof of Lemma 22 is the top triangle with its leg. We prove it as two results (mainly out of formalisation considerations)— the first form suffices by Lemma 10:





The proofs do not provide any insights and have been omitted.

4.5 Confluence

We have thus completed the proof of Lemma 22 and only one more lemma is needed before we can conclude our main result.

Lemma 30 $\dashrightarrow_{\alpha\cup\beta} \subseteq \dashrightarrow\!\!\!\!\!\twoheadrightarrow_{\alpha}; \dashrightarrow_{\mathrm{H}\beta} \subseteq \dashrightarrow\!\!\!\!\!\twoheadrightarrow_{\alpha\cup\beta}$

Proof By rule induction observing that both $\dashrightarrow\!\!\!\!\!\twoheadrightarrow_{\alpha}$ and $\dashrightarrow_{\mathrm{H}\beta}$ are reflexive. The proofs of the inclusions: $\dashrightarrow_{\beta} \subseteq \dashrightarrow_{\mathrm{H}\beta} \subseteq \dashrightarrow\!\!\!\!\!\twoheadrightarrow_{\beta}$, go through straightforwardly. □

Theorem 31 (Confluence of the Raw and Real λ-Calculi)

$$\mathit{Confl}(\dashrightarrow_{\alpha\cup\beta}) \wedge \mathit{Confl}(\rightarrow_{\beta}) \wedge \mathit{Confl}(\dashrightarrow_{\alpha^{\mathrm{C}}\cup\beta^{\mathrm{C}}}) \wedge \mathit{Confl}(\rightarrow_{\beta^{\mathrm{H}}})$$

Proof By Lemmas 5, 22, and 30 and then Theorem 16. □

[6] "Any two residuals of some sub-term in a residual of the original term are disjoint".

5 Conclusion

We have completed a confluence proof applying to several raw and real λ-calculi. It has been done by using first-order induction principles over Λ^{var} and reduction relations, only. It is the first proof we know of which clearly makes the raw-/real-calculi distinction. It does so by introducing a new result about preservation/reflection of confluence. It is also the first formalised equational result about a higher-order language which conducts its inductive reasoning over $\mathrm{FOAS}_{\mathcal{VN}}$, as you do informally by hand.

A Rational Reconstruction of the BVC We proved two results about parallel and completely developing β-reduction, Lemmas 27 and 28, in order to apply Takahashi's Trick. In summary, they say that irrespective of which pre-existing β-redexes in a BCF-term you contract in parallel and without performing renaming, it is possible to contract the residuals of the rest in parallel and without performing renaming and arrive at the completely developed term. All in all, the residual theory of $\dashrightarrow_\beta$ in λ^{var} is renaming-free up-to BCF-initiality. This is partly a consequence of Hyland's Disjointness Property [15] and partly due to our careful use of substitution. Said differently, Barendregt's moral:

> "**2.1.14.** Using 2.1.12/13 one can work with λ-terms the naive way."

is formally justifiable and is, in fact, an entirely reasonable way to conduct equational proofs about the λ-calculus when due care is taken to clarify the raw vs. real status of the established property.

References

1. Barendregt: *The Lambda Calculus — Syntax and Semantics.* North-Holland, 1984.
2. Burstall: Proving properties of programs by struct. ind. *Comp.J.*, 12, 1967.
3. Curry, Feys: *Combinatory Logic.* North-Holland, 1958.
4. de Bruijn: Lambda calculus notation with nameless dummies, a tool for auto. formula manipulation, with appl. to the CR Theorem. *Indag. Math.*, 34, 1972.
5. Despeyroux, Hirschowitz: HOAS with ind. in COQ. *LPAR*, 1994. LNAI 822.
6. Despeyroux, Pfenning, Schürmann: Prim. rec. for HOAS. *TLCA*, 1997. LNCS 1210.
7. Fiore, Plotkin, Turi: Abstract syntax and variable binding. In Longo [16].
8. Gabbay, Pitts: A new approach to abstract syntax involving binders. In Longo [16].
9. Girard: From the rules of logic to the logic of rules. To appear in *MSCS.*
10. Gordon, Melham: Five axioms of alpha-conversion. *TPHOL*, 1996. LNCS 1125.
11. Hindley: *The CR Prop. and a Result in Comb. Logic.* PhD thesis, Newcastle, 1964.
12. Hofmann: Semantical analysis of HOAS. In Longo [16].
13. Huet: Residual theory in λ-calculus: A formal development. *JFP*, 4(3), 1994.
14. Jouannaud, Kirchner: Compl. of a set of rules mod. a set of eq. *SIAM*, 15, 1986.
15. Klop: *Combinatory Reduction Systems.* Mathematical Centre Tracts 127, 1980.
16. Longo (ed.): *LICS-14*, 1999. IEEE Computer Society Press.
17. McKinna, Pollack: Some lambda calculus and TT formalized. To appear in *JAR.*
18. Nipkow: More CR proofs (in Isabelle/HOL). *CADE-13*, 1996. LNCS 1104.

19. Rose: Explicit substitution – tutorial & survey. BRICS-LS-96-13, 1996.
20. Rutten: A calc. of transition systems (towards univ. coalg.). CWI-CS-R9503, 1995.
21. David E. Schroer. *The Church-Rosser theorem.* PhD thesis, Cornell, June 1965.
22. Schürmann: *Automating the Meta Theory of Ded. Syst.* PhD thesis, CMU, 2000.
23. Shankar: A mechanical proof of the Church-Rosser Theorem. *J. ACM*, 35(3), 1988.
24. Takahashi: Parallel reductions in λ-calculus. *I. and C.*, 118, 1995.

A Commutative Diagrams

Formally, a commutative diagram is a set of vertices and a set of directed edges between pairs of vertices. A vertex is written as either $\bullet$ or $\circ$. Informally, this denotes quantification modes over terms, universal respectively existential. A vertex may be guarded by a predicate. Edges are written as the relational symbol they pertain to and are either full-coloured (black) or half-coloured (gray). Informally, the colour indicates assumed and concluded relations, respectively. An edge connected to a $\circ$ must be half-coloured. A diagram must be type-correct on domains. A property is read off of a diagram thus:

1. write universal quantifications for all $\bullet$s (over the relevant domains)
2. assume the full-coloured relations and the validation of any guard for a $\bullet$
3. conclude the guarded existence of all $\circ$s and their relations

The following diagram and property correspond to each other (for $\to \subseteq A \times A$).

$$\begin{array}{ll} (P)\ \bullet \longrightarrow \bullet & \forall e_1, e_2, e_3 \in A.\ e_1 \to e_2 \ \wedge\ e_1 \to e_3 \ \wedge\ P(e_1) \\ \quad\downarrow \quad\ \downarrow & \qquad\qquad\qquad\quad \Downarrow \\ \quad\bullet \longrightarrow \circ\,(Q) & \qquad\qquad\qquad\quad \exists e_4 \in A.\, e_2 \to e_4 \ \wedge\ e_3 \to e_4 \ \wedge\ Q(e_4) \end{array}$$

We will often leave quantification domains implicit and furthermore assume the standard disambiguating conventions for binding strength and associativity of connectives.