A Formalised First-Order Confluence Proof for the $\boldsymbol{\lambda}$-Calculus using One-Sorted Variable Names

René Vestergaard[1] and James Brotherston[2]

[1] CNRS-IML, Marseille, France, vester@iml.univ-mrs.fr*
[2] University of Edinburgh, Scotland, jjb@dai.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 proof principles of the standard syntax and the 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 of the λ-calculus coincides with Curry's and Hindley's when terms are considered equal up-to α and (ii) that the confluence properties of all considered systems 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 as follows:

$$\Lambda^{\mathrm{var}} ::= x \;\mid\; \Lambda^{\mathrm{var}}\Lambda^{\mathrm{var}} \;\mid\; \lambda x.\Lambda^{\mathrm{var}}$$

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 of a variable over a term, respectively. The variable names are implicitly taken to belong to a single infinite set of names, $\mathcal{VN}$, with a decidable equality relation (which canonically extends to the whole of Λ^{var}). Seen this way, Λ^{var} is first-order abstract syntax: $\mathrm{FOAS}^{\mathcal{VN}}$, or more generally FOAS, and as such comes equipped with a primitive, first-order principle of *structural induction* [2]:

$$\frac{\forall x.P(x) \quad \forall e_1, e_2.P(e_1) \wedge P(e_2) \to P(e_1e_2) \quad \forall x, e.P(e) \to P(\lambda x.e)}{\forall e.P(e)}$$

The Λ^{var}-equality is used implicitly in the premises of the induction principle as e_1, e_2, and e occur twice within their binding. Along with induction, the syntax also comes equipped with a primitive recursion principle which, incidentally,

* Supported under EU TMR grant # ERBFMRXCT-980170: LINEAR. Work done in part while visiting LFCS, University of Edinburgh from Heriot-Watt University.
** Work done with the support of a grant from LFCS, University of Edinburgh. Presently employed at CISA, University of Edinburgh.

also relies on Λ^{var}-equality. The recursion principle allows us to define auxiliary notions — like free variables — by case-splitting, aka recursive descent, over the terms. We will refer to structural induction, recursion, and equality as the primitive proof principles of Λ^{var}: $\text{PPP}(\Lambda^{\text{var}})$, or more generally PPP_{FOAS}.

On top of the terms we define *reduction relations*, as we shall see shortly. The main effect of reduction on Λ^{var} will be to insert arguments to abstractions into the places designated by the abstracted variable, i.e., to perform (higher-order) function application. Intuitively, we will therefore want to consider terms that only differ in the particular names used to express abstraction to be equal, so-called α-equivalence. On the technical side, we seemingly also need α-equivalence when "inserting arguments" to avoid unintended overlaps of variable names , see **The Issues** and Section 1.2 below. However, α-equivalence is a tricky construction as far as the algebra of the syntax goes and we will only undertake it after mature consideration. The problem is that α-equivalence breaks the Λ^{var}-equality underlying $\text{PPP}_{\text{FOAS}^{\mathcal{VN}}}$.

Although it is $\text{PPP}_{\text{FOAS}^{\mathcal{VN}}}$ that typically are said to be invoked when doing pen-and-paper proofs about the equational properties of the λ-calculus or of higher-order languages in general, those exact proof principles have so-far failed to underpin any corresponding formal proof development.[1] We resolve this open issue. Doing so is relevant both from a foundational and a practical perspective.

Foundationally it would be unsettling if our standard pen-and-paper proof practices could not be formalised, as has seemed to be the case till now. Such concerns have become highly pertinent by the large amount of alternative syntax formalisms that have been proposed recently [**?**]. Many of them, most noticeably de Bruijn [**?**], are explicitly justified by formalist (i.e., theorem proving) considerations

Practically, we show that pen-and-paper proof practices merely are incomplete (as opposed to incorrect) in a precise sense. To tighten them up to full formalist rigour, it suffices to introduce a fairly simple administrative proof layer on top of the two that are already there: key lemmas about the FOAS at hand and purely inductive reasoning about relations, seen abstractly. The fact that the former pen-and-paper proof layer typically carries over unchanged to a formal setting is by no means obvious. In fact, we consider substantiating this to be one of the main contributions of the paper.

Central to the administrative proof layer is a (the first, in fact) rational reconstruction of the widely used and very helpful but rather elusive Barendregt Variable Convention (BVC) [1].

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

[1] FIXME: Some come close by using induction over the size of terms.

To avoid any misunderstandings about what the BVC is or is not, we stress that it is uniformly invoked to justify the use of $\mathrm{PPP}_{\mathrm{FOAS}^{\mathcal{VN}}}$ while ignoring any enforced changes to variable names, cf. Section 1. While variable-name changes respect α-equivalence, they break Λ^{var}-equality and more generally $\mathrm{PPP}_{\mathrm{FOAS}^{\mathcal{VN}}}$, thus rendering uses of the BVC formally unjustified as it stands.

The Issues In $\mathrm{FOAS}^{\mathcal{VN}}$, name-overlaps seem inevitable when substituting a term into another, e.g., as part of a function application. If we, for example, wish to apply the function $\lambda x.(\lambda y.x)$ to some argument e, we cannot merely take $\lambda y.e$ to be the result because e could be, e.g., y. Whereas $\lambda y.x$ and $\lambda z.x$ intuitively have the same meaning (discard argument, return x), replacing x with a y lead to different results: the identity function versus discard argument, return y. 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[2] and, more problematically, (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:

$$\begin{tikzcd}
 & (\lambda y.\lambda x.xy)y \arrow[r, dashed, "\beta^{\mathrm{C}}"] & \lambda x.xy \\
(\lambda x.(\lambda y.\lambda x.xy)x)y \arrow[ru, dashed, "\beta^{\mathrm{C}}"] \arrow[rd, dashed, "\beta^{\mathrm{C}}"'] & & \\
 & (\lambda x.\lambda z.zx)y \arrow[r, dashed, "\beta^{\mathrm{C}}"'] & \lambda z.zy
\end{tikzcd}$$

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 (and we refer the reader to [?] for an example), the formal properties we can establish this way will require some interpretation — for details, we refer the reader to our Theorem 4 and the discussions surrounding it.

A Closer Look at First-Order Names Recent research has shown that there can be formalist advantages to employing a certain amount of ingenuity on the issue of variable names [8, 17].

In [8] and later articles, Gabbay and Pitts propose the use of so-called Fraenkel-Mostowski (FM) set theory to model syntax. Key to the proposal is ... permutation model of Zermelo-Fraenkel set theory with atoms, resulting in a notion of *set abstraction* that match the notion of function abstraction in FOAS. Gabbay and Pitts propose to base a theory of programming languages on $\mathrm{PPP}_{\mathrm{FM}^{\mathcal{VN}}}$. That said, t

It is not obvious what the exact correspondence between syntax based on FOAS and FM-sets is. For example, the exact form of variable names, the atoms underlying the FM-sets, becomes a non-trivial issues. For example, ... outlines that due to the provable failure of the axiom of choice in FM set theory, the

[2] This need not happen with parallel substitution [?] — thanks to Laurent Regnier for making the observation and to an anonymous referee for the reference.

addition of an axiom prescribing that a new, fresh variable name always can be found leads to an inconsistent framework.
FIXME: descriptions. Gabbay-Pitts use $\mathrm{PPP}_{\mathrm{FM}^{\mathcal{VN}}}$ but have not proved results such as λ-confluence. McKinna-Pollack is based on $\mathrm{FOAS}^{\mathcal{VN}\times\mathcal{VN}}$ but they do NOT use $\mathrm{PPP}_{\mathrm{FOAS}^{\mathcal{VN}\times\mathcal{VN}}}$. Instead they use ... rests on implicit comprehension.

A Closer Look at Syntax Representation The basic problems with $\mathrm{FOAS}^{\mathcal{VN}}$ has directly resulted in the inception of syntax formalisms (several of them recently) which overcome the issues by native means [4–8, 12]. In general, they mark a conceptual and formal departure from the naive qualities of $\mathrm{FOAS}^{\mathcal{VN}}$.

FIXME: descriptions.

A Closer Look at Derived Proof Principles for Λ^{var} FIXME: discussion of Gordon, Gordon-Melham, Foobar-Mason, Homeier.

Our Contribution We

- show that it is possible and feasible to conduct formal equational proofs about higher-order languages by simple, first-order means (i.e., PPP)
- show that this can be done over $\mathrm{FOAS}^{\mathcal{VN}}$, as done by hand
- formally justify informal proof practices, including 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

1.1 Terminology and Conventions

We say that a term *reduces* to another if the two are related by a *reduction* relation and we denote the relationship by an infix arrow between the two terms. The "direction" of the reduction should be thought of as being from-left-to-right. The sub-term of the left-hand side that a reduction step "acts upon" is called the *redex* of the reduction 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.*[3]

- An abstract rewrite systems, *ARS*, is a binary relation: $\rightarrow \subseteq A \times A$.
- The converse of a relation, $\rightarrow$, is written $(\rightarrow)^{-1}$.
- Composition is: $a \rightarrow_1 ; \rightarrow_2 c \;\Leftrightarrow^{\mathrm{def}}\; \exists b \,.\, a \rightarrow_1 b \,\wedge\, b \rightarrow_2 c$.
- Given two reduction relations $\rightarrow_1$ and $\rightarrow_2$, we have: $\rightarrow_{1\cup 2} =^{\mathrm{def}} \rightarrow_1 \cup \rightarrow_2$.

[3] This informal concept corresponds to a proper, typically inductive, formal notion.

– Any relation has a reflexive closure:[4]

$$\frac{e_1 \rightarrow e_2}{e_1 \multimap e_2} \qquad \frac{}{e \multimap e}$$

– Any relation has a reflexive, transitive closure:

$$\frac{e_1 \rightarrow e_2}{e_1 \twoheadrightarrow e_2} \qquad \frac{}{e \twoheadrightarrow e} \qquad \frac{e_1 \twoheadrightarrow e_2 \quad e_2 \twoheadrightarrow e_3}{e_1 \twoheadrightarrow e_3}$$

– Any relation has a reflexive, transitive, and symmetric closure:

$$\frac{e_1 \rightarrow_A e_2}{e_1 =_A e_2} \qquad \frac{}{e =_A e} \qquad \frac{e_1 =_A e_2 \quad e_2 =_A e_3}{e_1 =_A e_3} \qquad \frac{e_1 =_A e_2}{e_2 =_A e_1}$$

– A relation which is functional will be written with a based arrow: $\mapsto$.
– The situation of 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.*
– Two reduction steps *connect* if the end-term of one is the other's start-term.
– 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.
– *Residuals* are the descendants of terms under reduction [**?**,**?**].

Throughout this article, we will distinguish between *raw* and *real* calculi: inductive structures vs. the former factored by an equivalence. In order to tell them apart, for both technical and conceptual reasons, we use dashed respectively full-lined relational symbols to denote them. Generally, raw calculi have some measure of PPP whereas real calculi do not.

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.

[4] This and the next two items are immediately associated with primitive induction principles. Equality, however, is only point-wise (or extensional) and no recursion principle is possible. For an algebraic approach to rewriting (over an alternative notion of ARS) with a complete set of primitive proof principle, see [**?**].

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*, $\mathrm{FV}(-)$, see Section 3, Figure 1:

$$
\begin{aligned}
y\langle x := e\rangle &= \begin{cases} e & \text{if } x = y \\ y & \text{otherwise} \end{cases} \\
(e_1 e_2)\langle x := e\rangle &= e_1\langle x := e\rangle e_2\langle x := e\rangle \\
(\lambda y.e')\langle x := e\rangle &= \begin{cases} \lambda y.e' & \text{if } x = y \\ \lambda y.e'\langle x := e\rangle & \text{if } x \neq y \wedge (y \notin \mathrm{FV}(e) \vee x \notin \mathrm{FV}(e')) \\ \lambda z.e'\langle y := z\rangle\langle x := e\rangle & \text{o/w; } \textit{first } z \notin \{x\} \cup \mathrm{FV}(e) \cup \mathrm{FV}(e') \end{cases}
\end{aligned}
$$

Curry is seminal in giving a precise definition of substitution which takes into account that binding should be thought of as a syntactic notion: binding does not change while reducing, so to speak. In the above definition, this is expressed in the final clause which performs a binder renaming to prevent any free occurrences of y in e from becoming bound by the considered abstraction after the substitution is performed. Curry 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 10. 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])

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

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 17.

Schroer's Presentation FIXME: foobar.

FIXME: Where should this go? Surely not here.

1.3 Related Work

1.4 Acknowledgements

The first author wishes to thank Olivier Danvy, Jean-Yves Girard, Stefan Kahrs, Vincent van Oostrom, 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 of [?].

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 our homepages. At the time of writing, the confluence properties for our λ^{var}-calculus (Section 3) and the λ-calculus proper have been established (plus what is documented in [?]). 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. In particular, the proof developments consistently use left- or right-transitivity instead of proper reflexive, transitive induction as we do here. By, e.g., left-transitivity we mean:

$$\frac{}{e \twoheadrightarrow e} \qquad \frac{e_1 \rightarrow e_2 \quad e_2 \twoheadrightarrow e_3}{e_1 \twoheadrightarrow e_3}$$

We do so mainly for brevity but also to accommodate the indexed relations we will introduce shortly. The three are naturally equivalent in terms of expressivity.

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. The size of our proof scripts is in the order of 4000 lines of code; over 200 lemmas are proved in total during the development.

The second author's Honours dissertation contains more detailed information about the proof development itself and focuses in part on the automation issue. It is available from his homepage. 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 abstract rewriting methods we use. Some are new, some are well-known.

2.1 Preservation and Reflection of Confluence

Surprisingly, the results in this section appear 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.[5] 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[6] *from* $\rightarrow_A$ *to* $\rightarrow_B$ *if it is total and onto on points and a homomorphism from* $\rightarrow_A$ *to* $\rightarrow_B$*:*

$$\begin{tikzcd}
 & \bullet \arrow[d, mapsto, "\mathcal{M}"] & & \circ \arrow[d, mapsto, "\mathcal{M}"] & & \bullet \arrow[r, "A"'] \arrow[d, mapsto, "\mathcal{M}"'] \arrow[dr, phantom, "(homo)"] & \bullet \arrow[d, mapsto, "\mathcal{M}"] \\
(total) & \circ & (onto) & \bullet & & \bullet \arrow[r, "B"'] & \bullet
\end{tikzcd}$$

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:*[7]

[5] A special case of Theorem 4, 4 is reported in [14] and we contradict a result in [19].

[6] The name is inspired from [20].

[7] In the theorem, the notation $\not\rightarrow$ ($\not\leftrightarrow$) means existence of counter-examples.

1. $$\begin{tikzcd} \circ \arrow[r, "A"] \arrow[d, mapsto, "\mathcal{M}"'] & \circ \arrow[d, mapsto, "\mathcal{M}"] \\ \bullet \arrow[r, "B"] & \bullet \end{tikzcd}$$ $\Rightarrow \quad \diamond(\rightarrow_A) \not\Rightarrow \diamond(\rightarrow_B)$

2. $$\begin{tikzcd} \circ \arrow[r, "A"] \arrow[d, mapsto, "\mathcal{M}"'] & \circ \arrow[d, mapsto, "\mathcal{M}"] \\ \bullet \arrow[r, "B"] & \bullet \end{tikzcd}$$ $\Rightarrow \quad \diamond(\rightarrow_A) \not\Rightarrow \diamond(\rightarrow_B)$

3. $$\begin{tikzcd} \bullet \arrow[r, "A"] \arrow[d, mapsto, "\mathcal{M}"'] & \circ \arrow[d, mapsto, "\mathcal{M}"] \\ \bullet \arrow[r, "B"] & \bullet \end{tikzcd}$$ $\Rightarrow \quad \diamond(\rightarrow_A) \rightarrow \diamond(\rightarrow_B)$
$\wedge \ \diamond(\rightarrow_A) \not\leftarrow \diamond(\rightarrow_B)$

4. $$\begin{tikzcd} \bullet \arrow[r, "A"] \arrow[d, mapsto, "\mathcal{M}"'] & \bullet \arrow[d, mapsto, "\mathcal{M}"] \\ \bullet \arrow[r, "B"] & \bullet \end{tikzcd}$$ $\Rightarrow \quad \diamond(\rightarrow_A) \leftrightarrow \diamond(\rightarrow_B)$

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.

$$\begin{tikzcd} a_1 \arrow[r, "A"] \arrow[drr, dotted, mapsto] & a_2 \arrow[rr, dotted, mapsto] & & b_2 \\ & & b_1 \arrow[ur] \arrow[dr, "B"] & \\ a_1' \arrow[r, "A"'] \arrow[urr, dotted, mapsto] & a_2' \arrow[rr, dotted, mapsto] & & b_2' \end{tikzcd} \qquad \begin{tikzcd} & a_1 \arrow[dl] \arrow[dr] \arrow[rr, dotted, mapsto] & & b_1 \arrow[d, "B"] \\ a_2 \arrow[rrr, dotted, mapsto, bend right] & & a_3 \arrow[r, dotted, mapsto] & b_2 \end{tikzcd}$$

□

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 and even in most parts of the formal reasoning community when dealing with proof principles derived from FOAS^{Var} or with formalisms, such as HOAS, that have no native support for variable names although seemingly representing languages based on FOAS^{Var}. FIXME: Mention HOAS, Shankar, Gordon, Gordon-Melham, FooBar-Mason, Homeier. Outline problem with CR versus Confluence in rewriting modulo [Jouanneaud-Kirchner,Ohlenbusch].

2.2 The Abstract Proof Burden of Confluence

FIXME: Give due credit to Mitschke at the expense of Takahashi in the following: the idea of going for maximal resolution is borrowed from residual theory [Church-Rosser, Schroer] and was adapted into a secondary tool in [Mitschke]. Takahashi’s use of maximal resolution clearly is inpired from [Mitschke], in particular. However, it is a lot more general in Takahashi’s case in that Mitschke only uses it to address inner reductions.

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 $(\exists \rightarrow_2 . \rightarrow_1 \subseteq \rightarrow_2 \subseteq \twoheadrightarrow_1 \ \wedge\ \diamond(\rightarrow_2)) \ \Rightarrow\ \mathrm{Confl}(\rightarrow_1)$

Proof A formalisation is provided in [18] and is re-used here. $\square$

The point is that, since a parallel relation, $\rightarrow_2$ 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, $\rightarrow_a$ and $\rightarrow_b$, we have*

$$\begin{tikzcd}
(P)\bullet \arrow[d, "b"'] & & (Q)\bullet \arrow[dd, "b"'] \arrow[dr, "a"] & & & & (P \wedge Q)\bullet \arrow[dl, "a"'] \arrow[dr, "a"] & \\
\circ & \wedge & & \bullet \arrow[dl, "a"] & \Rightarrow & \bullet \arrow[dr, "a"'] & & \bullet \arrow[dl, "a"] \\
& & \bullet & & & & \circ &
\end{tikzcd}$$

Proof Straightforward. $\square$

The second premise is often referred to as a Triangle Property.

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. A prominent feature of the definition is the fact that the relations are renaming-free. Intuitively, one can think of our α- and β-relations as an orthonormal axiomatisation of the sought-after equational theory.

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

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

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

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 in Figure 2. Plain α-rewriting is given as: $e_1 \dashrightarrow_\alpha e_2 \overset{\mathrm{def}}{\Leftrightarrow} \exists y.e_1 \overset{y}{\dashrightarrow}_{i\alpha} e_2$*

The central point in the above definition is the use of side-conditions on the contraction rules in order to avert the need for binder-renaming. 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. Informally, the side-conditions express that the binders that must be passed in order to reach an actual substitution target may not capture any free variables in the term being substituted in. The side-conditions themselves coincide with the all-but-forgotten notion of *not free for.*

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 will refer to uses of the induction princples corresponding to the clauses of the relation definitions as *rule induction.* We also remind the reader that relation equality is extensional and that no recursion over relations is possible.

Substitution has been defined the way it has purely to enable us to prove certain "renaming sanity" properties for it. The following propostion thus establishes, in order: the identity substitution is indeed the identity, the voided substitution is indeed void, substitution is exhaustive when $\mathrm{Capt}_x(e) \cap \mathrm{FV}(e') = \emptyset$, and renaming with any non-free y is reversible. They, and more, are all needed in the Isabelle/HOL proof development.

Proposition 8 (Renaming Sanity) *For all $x, y \in \mathcal{VN}$ and $e, e' \in \Lambda^{\mathrm{var}}$:*

1. $e[x := x] = e$

$$\frac{y \notin \mathrm{Capt}_x(e) \cup \mathrm{FV}(e)}{\lambda x.e \overset{y}{\dashrightarrow}_{i\alpha} \lambda y.e[x := y]}\,(i\alpha) \qquad \frac{e \overset{y}{\dashrightarrow}_{i\alpha} e'}{\lambda x.e \overset{y}{\dashrightarrow}_{i\alpha} \lambda x.e'}\,(\mathrm{L}_{i\alpha})$$

$$\frac{e_1 \overset{y}{\dashrightarrow}_{i\alpha} e_1'}{e_1e_2 \overset{y}{\dashrightarrow}_{i\alpha} e_1'e_2}\,(\mathrm{Al}_{i\alpha}) \qquad \frac{e_2 \overset{y}{\dashrightarrow}_{i\alpha} e_2'}{e_1e_2 \overset{y}{\dashrightarrow}_{i\alpha} e_1e_2'}\,(\mathrm{Ar}_{i\alpha})$$

$$\frac{\mathrm{FV}(e_2) \cap \mathrm{Capt}_x(e_1) = \emptyset}{(\lambda x.e_1)e_2 \dashrightarrow_\beta e_1[x := e_2]}\,(\beta) \qquad \frac{e \dashrightarrow_\beta e'}{\lambda x.e \dashrightarrow_\beta \lambda x.e'}\,(\mathrm{L}_\beta)$$

$$\frac{e_1 \dashrightarrow_\beta e_1'}{e_1e_2 \dashrightarrow_\beta e_1'e_2}\,(\mathrm{Al}_\beta) \qquad \frac{e_2 \dashrightarrow_\beta e_2'}{e_1e_2 \dashrightarrow_\beta e_1e_2'}\,(\mathrm{Ar}_\beta)$$

Fig. 2. Raw α- (indexed) and β-reduction.

$$\begin{array}{rlrl}
\mathrm{BV}(x) &= \emptyset & \mathrm{UB}(x) &= \mathrm{True} \\
\mathrm{BV}(e_1e_2) &= \mathrm{BV}(e_1) \cup \mathrm{BV}(e_2) & \mathrm{UB}(e_1e_2) &= \mathrm{UB}(e_1) \wedge \mathrm{UB}(e_2) \wedge \mathrm{BV}(e_1) \cap \mathrm{BV}(e_2) = \emptyset \\
\mathrm{BV}(\lambda x.e) &= \mathrm{BV}(e) \cup \{x\} & \mathrm{UB}(\lambda x.e) &= \mathrm{UB}(e) \wedge x \notin \mathrm{BV}(e)
\end{array}$$

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

2. $x \notin \mathrm{FV}(e) \;\Rightarrow\; e[x := e'] = e$
3. $x \notin \mathrm{FV}(e') \;\wedge\; (\mathrm{Capt}_x(e) \cap \mathrm{FV}(e') = \emptyset) \;\Rightarrow\; x \notin \mathrm{FV}(e[x := e'])$
4. $y \notin \mathrm{FV}(e) \;\Rightarrow\; e[x := y][y := x] = e$

Proof All proofs are straightforward structural inductions in e. We show the details of the second property to highlight what this means specifically. We remark that Isabelle/HOL proves the results fully automatically after being instructed to proceed by induction on e.

We aim to prove: $x \notin \mathrm{FV}(e) \Rightarrow e[x := e'] = e$, and thus assume $x \notin \mathrm{FV}(e)$.

Case $e \equiv y$**:** By the assumption and the definition of free variables, we immediately conclude $y \neq x$. By unravelling the definition of substitution: $y[x := e'] = y$, we are done.

Case $e \equiv e_1e_2$**:** By definition of substitution $(e_1e_2)[x := e'] = e_1[x := e']e_2[x := e']$. As $x \notin \mathrm{FV}(e_1e_2) \;\Leftrightarrow\; x \notin \mathrm{FV}(e_1) \wedge x \notin \mathrm{FV}(e_2)$, we can apply the induction hypothesis twice: $e_i[x := e']{=}e_i$ for $i = 1, 2$, and we are done.

Case $e \equiv \lambda z.e_0$**:** We case-split on z.

Case $z = x$ We are immediately done by definition of substitution.

Case $z \neq x \wedge z \notin \mathrm{FV}(e')$**:** By $x \notin \mathrm{FV}(\lambda z.e_0)$, $z \neq x$, and the definition of free variables, we have $x \notin \mathrm{FV}(e_0)$. We can therefore apply the induction hypothesis to the unravelling of the definition of substitution to get $(\lambda z.e_0)[x := e'] = \lambda z.e_0[x := e'] = \lambda z.e_0$.

Case $z \neq x \wedge z \in \mathrm{FV}(e')$**:** We are immediately done by unravelling the definition of substitution: $(\lambda z.e_0)[x := e'] = \lambda z.e_0$!

□

We stress that the last clause of the proof (which "incorrectly" discards the substitution) merely goes to show an algebraic property of the defined notion of substitution. In actual uses of substitution, the clause will never be invoked. In fact, we show next that as far as our use of substitution is concerned, our notion overlaps with Curry's.

Proposition 9 $\mathrm{FV}(e_2) \cap \mathrm{Capt}_x(e_1) = \emptyset \;\Rightarrow\; e_1[x := e_2] = e_1\langle x := e_2\rangle$

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 10 $\dashrightarrow_{\alpha^{\mathrm{H}}} \subseteq \dashrightarrow_{\alpha} \subseteq (\dashrightarrow_{\alpha^{\mathrm{C}}})^{-1}$

Proof The reasoning in the first case is analogous to the second which we show; it is done by a rule induction in the $\overset{y}{\dashrightarrow}_{i\alpha}$-relation underlying $\dashrightarrow_{\alpha}$.

Case $(i\alpha)$**:** The premise of the considered rule prescribes, amongst other things, that $y \notin \mathrm{FV}(e)$ for given y and $\lambda x.e$. By definition we therefore have $\lambda x.e\ (\dashrightarrow_{\alpha^{\mathrm{C}}})^{-1}\ \lambda y.e\langle x := y\rangle$. As the premise of the rule allows us to invoke Proposition **??**: $e[x := y] = e\langle x := y\rangle$, we are done.

Cases $(\mathrm{L}_{i\alpha})$, $(\mathrm{Al}_{i\alpha})$, $(\mathrm{Ar}_{i\alpha})$**:** Trivial.

□

Lemma 11 ($\dashrightarrow_{\alpha}$-Symmetry) $$\begin{tikzcd}\bullet \arrow[r, dashed, "\alpha"] & \bullet \arrow[l, dashed, bend left, "\alpha"]\end{tikzcd}$$

Proof By a straightforward rule induction, using Proposition **??**, **??**. □

Lemma 12 $\aleph\ = \dashrightarrow\!\!\!\!\rightarrow_{\alpha} = \;==_{\alpha}$

Proof From Lemmas 1 and 10 respectively Lemma 11. □

Lemma 13 $\dashrightarrow_{\beta} \subseteq \dashrightarrow_{\beta^{\mathrm{C}}} \subseteq \dashrightarrow\!\!\!\!\rightarrow_{\alpha}; \dashrightarrow_{\beta}$

Proof The first inclusion follows from Proposition 9. 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 12, they can thus be expressed by $\mathrel{-\,-\!\!\twoheadrightarrow}_{\alpha}$. It suffices to observe that no renaming is performed following the "passing" of the substitution invoked by the β-rule. $\square$

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

Definition 14 (The Real λ-Calculus)

- $\Lambda = \Lambda^{\mathrm{var}} / =\!=_{\alpha}$
- $\begin{array}{rcl} \lfloor - \rfloor : \Lambda^{\mathrm{var}} & \longrightarrow & \Lambda \\ e & \mapsto & \{e' \mid e =\!=_{\alpha} e'\} \end{array}$
- $\lfloor e_1 \rfloor \rightarrow_{\beta} \lfloor e_2 \rfloor \overset{def}{\Leftrightarrow} e_1 =\!=_{\alpha}; \dashrightarrow_{\beta}; =\!=_{\alpha} e_2$[8]

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

Proposition 15 (Point-wise Equivalence)

$$\lfloor e \rfloor \twoheadrightarrow_{\beta} \lfloor e' \rfloor \quad \Leftrightarrow \quad e\,(=\!=_{\alpha}; \dashrightarrow_{\beta}; =\!=_{\alpha})^{\star}\, e' \;\vee\; e =\!=_{\alpha} e'$$

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. $\square$

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

Lemma 16 (Point-wise Equivalence under Structural Symmetry)

$$\lfloor e \rfloor \twoheadrightarrow_{\beta} \lfloor e' \rfloor \Leftrightarrow e \mathrel{-\,-\!\!\twoheadrightarrow}_{\alpha\cup\beta} e' \Leftrightarrow e \mathrel{-\,-\!\!\twoheadrightarrow}_{\alpha^{\mathrm{C}}\cup\beta^{\mathrm{C}}} e' \Leftrightarrow \lfloor e \rfloor_{\mathrm{H}} \twoheadrightarrow_{\beta^{\mathrm{H}}} \lfloor e' \rfloor_{\mathrm{H}}$$

Proof From Lemma 11, it is trivial to see that $(=\!=_{\alpha}; \dashrightarrow_{\beta}; =\!=_{\alpha})^{\star} \cup =\!=_{\alpha} = \mathrel{-\,-\!\!\twoheadrightarrow}_{\alpha\cup\beta}$ and the first biimplication is established by Proposition 15. The second biimplication follows by Lemmas 10, 12, and 13. The last biimplication follows in an analogous manner. $\square$

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 17 (Equational and Confluence Equivalence)

[8] This definition is equivalent to the obvious inductive one: $\dfrac{e_1 \dashrightarrow_{\beta} e_2}{\lfloor e_1 \rfloor \rightarrow_{\beta} \lfloor e_2 \rfloor}$.

$$\frac{}{x \mathrel{-\!\!\Vert\!\!\rightarrow}_\beta x} \qquad \frac{e \mathrel{-\!\!\Vert\!\!\rightarrow}_\beta e'}{\lambda x.e \mathrel{-\!\!\Vert\!\!\rightarrow}_\beta \lambda x.e'} \qquad \frac{e_1 \mathrel{-\!\!\Vert\!\!\rightarrow}_\beta e_1' \quad e_2 \mathrel{-\!\!\Vert\!\!\rightarrow}_\beta e_2'}{e_1e_2 \mathrel{-\!\!\Vert\!\!\rightarrow}_\beta e_1'e_2'}$$

$$\frac{e_1 \mathrel{-\!\!\Vert\!\!\rightarrow}_\beta e_1' \quad e_2 \mathrel{-\!\!\Vert\!\!\rightarrow}_\beta e_2' \quad \mathrm{FV}(e_2') \cap \mathrm{Capt}_x(e_1') = \emptyset}{(\lambda x.e_1)e_2 \mathrel{-\!\!\Vert\!\!\rightarrow}_\beta e_1'[x := e_2']}$$

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

$$\frac{}{x \mathrel{\dashrightarrow\!\!|}_\beta x} \qquad \frac{e \mathrel{\dashrightarrow\!\!|}_\beta e'}{\lambda x.e \mathrel{\dashrightarrow\!\!|}_\beta \lambda x.e'} \qquad \frac{e \mathrel{\dashrightarrow\!\!|}_\beta e'}{xe \mathrel{\dashrightarrow\!\!|}_\beta xe'} \qquad \frac{e_1e_2 \mathrel{\dashrightarrow\!\!|}_\beta e' \quad e_3 \mathrel{\dashrightarrow\!\!|}_\beta e_3'}{(e_1e_2)e_3 \mathrel{\dashrightarrow\!\!|}_\beta e'e_3'}$$

$$\frac{e_1 \mathrel{\dashrightarrow\!\!|}_\beta e_1' \quad e_2 \mathrel{\dashrightarrow\!\!|}_\beta e_2' \quad \mathrm{FV}(e_2') \cap \mathrm{Capt}_x(e_1') = \emptyset}{(\lambda x.e_1)e_2 \mathrel{\dashrightarrow\!\!|}_\beta e_1'[x := e_2']}$$

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

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

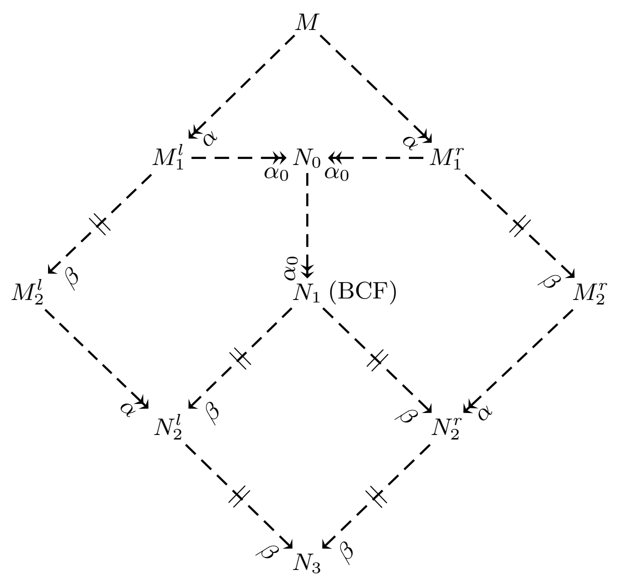
Proof The first result is immediate following Lemma 16. As for the second result, the definitional totality and surjectivity of $\lfloor - \rfloor$ and $\lfloor - \rfloor_{\mathrm{H}}$ combined with Lemma 16 allow us to apply Theorem 4, case 4 repeatedly. □

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 A Raw Diamond Property and λ-Confluence

As suggested by Sections 2 and 3, we are searching for 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. Justified by the fact that we are reasoning formally, we use Takahashi's method of defining a one-step parallel relation directly by induction over terms. The benefit is of course that we, by doing so, get direct access to the exact inductive proof principle we need for showing the diamond property we are after.

Definition 18 Parallel *β-reduction, $\mathrel{-\!\!\Vert\!\!\rightarrow}_\beta$, is defined in Figure 4.*



$$\begin{tikzcd}
 & & & M \arrow[dl, dashed, two heads, "\alpha"'] \arrow[dr, dashed, two heads, "\alpha"] & & & \\
 & & M_1^l \arrow[r, dashed, two heads, "\alpha_0"'] \arrow[dl, dashed, "\beta"', "\nparallel" description] & N_0 \arrow[dd, dashed, two heads, "\alpha_0"'] & M_1^r \arrow[l, dashed, two heads, "\alpha_0"] \arrow[dr, dashed, "\beta", "\nparallel" description] & & \\
 & M_2^l \arrow[dr, dashed, two heads, "\alpha"'] & & & & M_2^r \arrow[dl, dashed, two heads, "\alpha"] & \\
 & & & N_1 \text{ (BCF)} \arrow[dl, dashed, "\beta", "\nparallel" description] \arrow[dr, dashed, "\beta"', "\nparallel" description] & & & \\
 & & N_2^l \arrow[dr, dashed, "\beta"', "\nparallel" description] & & N_2^r \arrow[dl, dashed, "\beta", "\nparallel" description] & & \\
 & & & N_3 & & &
\end{tikzcd}$$

Fig. 6. Foo bar

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.

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 in some cases. For now, we merely present:

Lemma 20 $\dashrightarrow\!\!|_{\beta} \subseteq \mathrel{-\!\!\nparallel\!\!\rightarrow}_{\beta}$

Proof Straightforward. $\square$

The Overall Proof Structure Having thus established the basics, we outline the proof of the diamond property of the following relation: $\dashrightarrow\!\!\!\!\!\twoheadrightarrow_{\alpha};\mathrel{-\!\!\nparallel\!\!\rightarrow}_{\beta}$, before supplying the actual details of the proof. In order to use the BVC in our proof, we first present it as a predicate on Λ^{var}, cf. Figure 3.

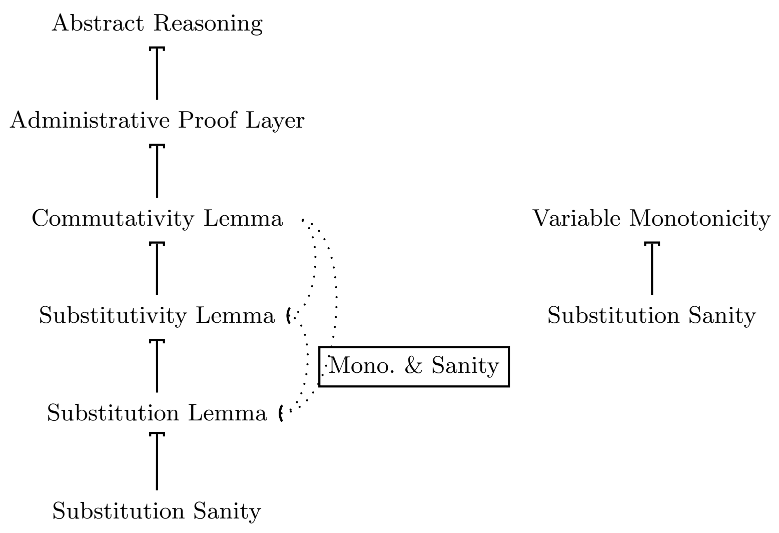


Fig. 7. The proof-layer hierarchy for equational reasoning about λ over $\mathrm{FOAS}^{\mathrm{Var}}$. The square up-arrows read "is the key lemma for a main case of" whereas the rounded, dotted down-arrows are "invokes by using variable monotonicity and substitution sanity to justify the associated side-conditions".

Definition 21 (Barendregt Conventional Form)

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

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

Proof For the diverging Ms given, we can construct the resolving Ns in Figure **??** in order. The ensuing sections will detail the individual diagrams. The $\dashrightarrow_{\alpha_0}$-relation is introduced in Definition 26 as the fresh-naming restriction of $\dashrightarrow_{\alpha}$. It serves to facilitate the commutativity with β on either side of the diagram. $\square$

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. However, it also means that the usual key lemma and its proof when doing a pen-and-paper β-confluence proof *à la* FooBarBaz is used directly in our proof. FIXME: clean up, elaborate.

4.1 The Proof-Layer Hierarchy

The results we need for the full confluence proof can be separated into their different levels of abstraction, with algebraic properties of the calculus belonging to the lowest level and abstract rewriting properties to the highest. The resulting proof layer hierarchy is shown in Figure 7. Typically, the key part of the proof burden for a result belonging to any proof layer is resolved by lemmas belonging to the layer immediately above it; the results in the Substitution Sanity class are proved directly by structural induction (cf. Proposition 8). The hierarchy also contains *feedback loops* — denoted in the diagram by dotted downwards arrows — which arise when a key lemma for a result has associated side conditions on its application. In such cases these conditions can usually be substantiated by the application of lemmas from the Sanity and Monotonicity classes. This is so because the side conditions are specified at the same level of abstraction as Substitution Sanity, which is to say they specify algebraic properties of sets of variables.

Substitution Sanity results specify behaviour of terms under substitution, whereas Variable Monotonicity results specify the behaviour of sets of variables under reduction. Their purpose when applied in a feedback loop is to justify the side-conditions on a particular key lemma by extracting the relevant low-level variable information from the higher-level premises. The Monotonicity result most often needed is the following:

Proposition 23 (Parallel β Variable Monotonicity)

$$e \mathrel{\text{-H}\!\!\to_\beta} e' \;\Rightarrow\; \mathrm{FV}(e') \subseteq \mathrm{FV}(e) \;\wedge\; \mathrm{BV}(e') \subseteq \mathrm{BV}(e)$$

Proof By rule induction in the $\mathrel{\text{-H}\!\!\to_\beta}$-reduction with the only non-trivial case following by the application of Substitution Sanity. $\square$

Substitution Lemmas are concerned with commutativity of (well-behaved) substitutions, while Substitutivity results show that reductions respect (well-behaved) substitution. They are non-trivial to prove formally and several of each are required in the proof development. 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 24 (Substitution)

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

Proof By structural induction in e_1 and an exhaustive splitting on the (many) different substitution cases. Some cases require Substitution Sanity in order to go through. $\square$

Lemma 25 (Parallel β Substitutivity)

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

Proof By rule induction on $e_1 \;{-}\!\!\Vert\!\!\rightarrow_\beta\; e_1'$. The reduction case splits further into three subcases, each requiring the use of a Substitution Lemma. The associated side conditions are resolved by using Monotonicity and Substitution Sanity. □

We refer the interested reader to the complete Isabelle/HOL proof development at our homepages 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 is given point-wise but has a straightforward inductive equivalent.

Definition 26 $e \dashrightarrow_{\alpha_0} e' \;\Leftrightarrow^{\mathrm{def}}\; \exists z. e \overset{z}{\dashrightarrow}_{i\alpha} e' \wedge z \notin \mathrm{FV}(e) \cup \mathrm{BV}(e)$

Lemma 27

$$\begin{tikzcd}
\bullet \arrow[r, dashed, "\beta", "\Vert" marking] \arrow[d, dashed, two heads, "\alpha_0"'] & \bullet \arrow[d, dashed, gray, two heads, "\alpha"] \\
\bullet \arrow[r, dashed, gray, "\beta", "\Vert" marking] & \circ
\end{tikzcd}$$

Proof By reflexive-transitive induction in $\dashrightarrow\!\!\!\!\rightarrow_{\alpha_0}$. The reflexive and transitive cases are trivial. Now consider the base case:

$$\begin{tikzcd}
\bullet \arrow[r, dashed, "\beta", "\Vert" marking] \arrow[d, dashed, "\alpha_0"'] & \bullet \arrow[d, dashed, gray, two heads, "\alpha"] \\
\bullet \arrow[r, dashed, gray, "\beta", "\Vert" marking] & \circ
\end{tikzcd}$$

This property follows by rule induction in $\overset{y}{\dashrightarrow}_{i\alpha_0}$ and then an involved case-splitting on ${-}\!\!\Vert\!\!\rightarrow_\beta$; the details are substantial and are omitted here. The proof relies crucially on the fact that y is fresh and furthermore takes advantage of the fact that it suffices to use the same y at each step in the $\dashrightarrow\!\!\!\!\rightarrow_{i\alpha}$-resolution. □

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 28 (BCF) $\begin{tikzcd} \bullet \arrow[r, dashed, gray, "\beta"', ->|] & \circ \end{tikzcd}$

Proof By structural induction using Proposition 23 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 29

$$\begin{tikzcd}
\bullet \arrow[rr, dashed, "\beta"] \arrow[dr, "\parallel" marking, "\beta"'] & & \bullet \\
& \bullet \arrow[ur, gray, "\parallel" marking, "\beta"'] &
\end{tikzcd}$$

Proof By rule induction in $\dashrightarrow\!\!|_\beta$ using Lemma 25. □

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 $\dashrightarrow\!\!|_\beta$ in any non-trivial cases. 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].[9]

Lemma 30

$$\begin{tikzcd}
(\mathrm{BCF})\ \bullet \arrow[r, dashed, "\parallel" marking, "\beta"] \arrow[d, dashed, "=" marking, "\beta"'] & \bullet \arrow[d, gray, dashed, "=" marking, "\beta"] \\
\bullet \arrow[r, gray, dashed, "\parallel" marking, "\beta"] & \circ
\end{tikzcd}$$

Proof From Lemmas 28 and 29 by using Takahashi's Trick, Lemma 6. □

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

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. Intuitively the proof is easy: two α-equivalent terms have the same "structure" so it suffices to find enough fresh variable names and rename with them in the same order for the two terms. Formally, however, there are a number of problems with this approach, not least in trying to quantify the notion of "structure" as well as in coming up with formal proof principles that can be applied to it. The approach we take instead is to proceed as dictated by the proof principles we have introduced. The proofs might not provide much insight into the overall confluence proof we are presenting but they do show-case what rule and structural induction really means.

The biggest technical problem we encounter is that we have neither $\diamond(\dashrightarrow_{\alpha_0})$ nor $\diamond(\dashrightarrow\!\circ_{\alpha_0})$, and thus cannot conclude the needed $\diamond(\dashrightarrow\!\!\!\!\rightarrow_{\alpha_0})$ in a straightforward manner. A counter-example is the following, with x, y, and z all different:

$$\begin{tikzcd}[row sep=tiny]
 & \lambda z.\lambda x.xz \\
\lambda y.\lambda x.xy \arrow[ur, dashed, "i\alpha_0"] \arrow[dr, dashed, "i\alpha_0"'] & \\
 & \lambda y.\lambda z.zy
\end{tikzcd}$$

The problem is that both steps use the fresh z but in different positions. We therefore necessarily have to proceed while doing low-level reasoning over the indexes. A slightly anomalous indexing scheme for composed versions of α_0 turns out to be useful in order to do so. It allows us to retain an index in the reflexive case, i.e., when performing an "empty" step. It uses lists of indexes: $\vec{z_i}$. We write z for the list with one element, $\{z_i\}$ for the set of elements in a list, juxtaposition for list extension, and $||-||$ for the length of a list.

Definition 31 (Index-Anomalous Reflexive, Transitive $i\alpha_0$) [10]

$$\frac{e_1 \overset{z}{\dashrightarrow}_{i\alpha_0} e_2}{e_1 \overset{z}{\dashrightarrow\!\!\!\!\rightarrow}_{i\alpha_0} e_2} \qquad \frac{z \notin \mathrm{FV}(e) \cup \mathrm{BV}(e)}{e \overset{z}{\dashrightarrow\!\!\!\!\rightarrow}_{i\alpha_0} e} \qquad \frac{e_1 \overset{\vec{z_i}}{\dashrightarrow\!\!\!\!\rightarrow}_{i\alpha_0} e_2 \qquad e_2 \overset{\vec{z_j}}{\dashrightarrow\!\!\!\!\rightarrow}_{i\alpha_0} e_3}{e_1 \overset{\vec{z_i}\vec{z_j}}{\dashrightarrow\!\!\!\!\rightarrow}_{i\alpha_0} e_3}$$

Lemma 32 (Quasi-Confl($\dashrightarrow_{\alpha_0}$)) *For $\vec{z_i}$ and $\vec{z_j}$ such that $\{z_i\} \cap \{z_j\} = \emptyset$:*

$$\begin{tikzcd}
\bullet \arrow[r, dashed, two heads, "i\alpha_0", "\vec{z_j}"'] \arrow[d, dashed, two heads, "i\alpha_0"', "\vec{z_i}"] & \bullet \arrow[d, dashed, two heads, "i\alpha_0", "\vec{z_i}"'] \\
\bullet \arrow[r, dashed, two heads, "\vec{z_j}", "i\alpha_0"'] & \circ
\end{tikzcd}$$

Proof By two nested reflexive-transitive inductions. The only non-trivial case is the following, for any $z_1 \neq z_2$:

$$\begin{tikzcd}
\bullet \arrow[r, dashed, "i\alpha_0", "z_2"'] \arrow[d, dashed, "i\alpha_0"', "z_1"] & \bullet \arrow[d, dashed, "i\alpha_0", "z_1"'] \\
\bullet \arrow[r, dashed, -{Circle[open]}, "z_2", "i\alpha_0"'] & \circ
\end{tikzcd}$$

[10] Reflexive closure, $\overset{z}{\dashrightarrow\!\circ}_{i\alpha_0}$, is defined analogously.

This property is proved by rule induction in $\overset{z}{\dashrightarrow}_{i\alpha_0}$. Every case except the variable case requires intricate reasoning with names; reflexivity is needed for the case where the divergence is caused by two α_0-steps on the same abstraction. □

Lemma 33 (Fresh-Naming Confl($\dashrightarrow_\alpha$))

$$\begin{tikzcd}
\bullet \arrow[r, dashed, two heads, "\alpha"] \arrow[d, dashed, two heads, "\alpha"'] & \bullet \arrow[d, dashed, two heads, "\alpha_0"] \\
\bullet \arrow[r, dashed, two heads, "\alpha_0"'] & \circ
\end{tikzcd}$$

Proof By Lemma **??**, it suffices to prove:

$$\begin{tikzcd}
\bullet \arrow[rr, dashed, two heads, "\alpha"] \arrow[dr, dashed, two heads, "\alpha_0"'] & & \bullet \arrow[dl, dashed, two heads, "\alpha_0"] \\
& \circ &
\end{tikzcd}$$

The proof is a reflexive, transitive induction in $\overset{\vec{x_i}}{\dashrightarrow\!\!\!\!\rightarrow}_{i\alpha}$. The base case asserts, for any $x \neq z$, that

$$\begin{tikzcd}
\bullet \arrow[rr, dashed, "i\alpha", "x"'] \arrow[dr, dashed, "z", "i\alpha_0"'] & & \bullet \arrow[dl, dashed, "z"', "i\alpha_0"] \\
& \circ &
\end{tikzcd}$$

The proof is straightforward. The reflexive case is trivial whereas the transitive case is proved as follows

$$\begin{tikzcd}
M_1 \arrow[rr, dashed, two heads, "\vec{x_i}\ i\alpha"] \arrow[dr, dashed, two heads, "i\alpha_0", "\vec{z_i}"'] & & M_2 \arrow[rr, dashed, two heads, "\vec{x_j}\ i\alpha"] \arrow[dl, dashed, two heads, "i\alpha_0"', "\overleftarrow{z_i}"] \arrow[dr, dashed, two heads, "i\alpha_0", "\vec{z_j}"'] & & M_3 \arrow[dl, dashed, two heads, "i\alpha_0"', "\overleftarrow{z_j}"] \\
& N_1 \arrow[dr, dashed, two heads, "i\alpha_0", "\vec{z_j}"'] & & N_2 \arrow[dl, dashed, two heads, "i\alpha_0"', "\overleftarrow{z_i}"] & \\
& & N_3 & &
\end{tikzcd}$$

The quantification of the above variables states that $\{x_i, x_j\}$, $\{z_i\}$, and $\{z_j\}$ all are pairwise disjoint. The upper triangles exist by induction hypothesis. The lower diamond is Lemma **??**. Notice the reversal of the resolving indexes. □

The final result we need to prove is the existence of the "leg" in the proof of Lemma **??**.

Lemma 34 (Existence of α_0-Renaming Sequence to BCF)

$$\begin{tikzcd}
\bullet \arrow[r, dashed, two heads, "\alpha_0"] & \underset{(BCF)}{\circ}
\end{tikzcd}$$

Proof The key result in order to show this lemma is the following — we write $\#_\lambda(e)$ for the number of λ-abstractions in e:

$$\begin{array}{l}\forall\, \vec{z_i}, e_1.\ \|\, \vec{z_i}\, \| = \#_\lambda(e_1)\ \wedge\ \{z_i\}\text{ all different}\ \wedge\ (\{z_i\} \cap (\mathrm{FV}(e_1) \cup \mathrm{BV}(e_1)) = \emptyset) \\ \Downarrow \\ \exists e_2.e_1 \overset{\vec{z_i}}{\dashrightarrow\!\!\!\twoheadrightarrow}_{i\alpha_0} e_2\ \wedge\ \mathrm{BCF}(e_2)\ \wedge\ \{z_i\} = \mathrm{BV}(e_2)\end{array}$$

The result follows by a lengthy structural induction in e_1. □

To see why we need all the variable information in the listed property, consider the case of a naive proof by structural induction where e_1 is an application: $e'e''$; knowing that $e' \dashrightarrow\!\!\!\twoheadrightarrow_{\alpha_0} e_1'$, $e'' \dashrightarrow\!\!\!\twoheadrightarrow_{\alpha_0} e_1''$ and $\mathrm{BCF}(e_1')$ & $\mathrm{BCF}(e_1'')$ does *not* enable us to conclude $\mathrm{BCF}(e_1'e_1'')$ and we are stuck.

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 35 $\dashrightarrow_{\alpha\cup\beta} \subseteq \dashrightarrow\!\!\!\twoheadrightarrow_\alpha;\ \mathrel{-\!\mathtt{H}\!\!\to}_\beta \subseteq \dashrightarrow\!\!\!\twoheadrightarrow_{\alpha\cup\beta}$

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

Theorem 36 (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 35 and then Theorem 17. □

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 proceeds by $\mathrm{PPP}_{\mathrm{FOAS}^{\mathcal{VN}}}$, as done informally by hand.

A Rational Reconstruction of the BVC We proved two results about parallel and completely developing β-reduction, Lemmas 28 and 29, 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 $\rightarrow \subseteq A \times A$).

$$\begin{tikzcd}
(P)\, \bullet \arrow[r] \arrow[d] & \bullet \arrow[d, gray] \\
\bullet \arrow[r, gray] & \circ\, (Q)
\end{tikzcd}$$

$$\begin{array}{c}
\forall e_1, e_2, e_3 \in A\,.\; e_1 \rightarrow e_2 \;\wedge\; e_1 \rightarrow e_3 \;\wedge\; P(e_1) \\
\Downarrow \\
\exists e_4 \in A\,.\, e_2 \rightarrow e_4 \;\wedge\; e_3 \rightarrow e_4 \;\wedge\; Q(e_4)
\end{array}$$

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