Electronic Notes in Theoretical Computer Science 58 No. 1 (2001)
URL: http://www.elsevier.nl/locate/entcs/volume58.html 19 pages

The Mechanisation of Barendregt-Style Equational Proofs (the Residual Perspective)

René Vestergaard [1,2]

Institut de Mathématiques de Luminy
Centre National de la Recherche Scientifique
Marseille, France

James Brotherston [3,4]

Centre for Intelligent Systems and Applications
University of Edinburgh
Edinburgh, Scotland, UK

Abstract

We show how to mechanise equational proofs about higher-order languages by using the primitive proof principles of first-order abstract syntax over one-sorted variable names. We illustrate the method here by proving (in Isabelle/HOL) a technical property which makes the method widely applicable for the λ-calculus: the residual theory of β is renaming-free up-to an initiality condition akin to the so-called Barendregt Variable Convention. We use our results to give a new diagram-based proof of the development part of the strong finite development property for the λ-calculus. The proof has the same equational implications (e.g., confluence) as the proof of the full property but without the need to prove SN. We account for two other uses of the proof method, as presented elsewhere. One has been mechanised in full in Isabelle/HOL.

1 Introduction

There is a de facto difference between pen-and-paper and mechanised proof practices for equational properties of higher-order languages, i.e., languages

[1] Supported under EU TMR grant # ERBFMRXCT-980170: LINEAR.
[2] Email: vester@iml.univ-mrs.fr, WWW: http://iml.univ-mrs.fr/~vester
[3] Work undertaken while supported by LFCS, Edinburgh.
[4] Email: jjb@dai.ed.ac.uk, WWW: http://www.dai.ed.ac.uk/~jjb

with binding (aka abstraction). The point of disagreement is one of syntactic representation rather than any specifics of proof methodologies. As for the pen-and-paper practices, first-order abstract syntax with one-sorted variable names is a rather simple formalism that is well-suited to focus a reader's attention on the sentiments or insights being communicated in a proof [6,16,17,25]. This is partly so because the syntax comes equipped with particularly simple primitive induction and recursion principles [3]. Unfortunately, the variable names used to express binding can clash when reducing terms. 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, [5] and (ii) as a term typically can reduce in several ways, the resulting name for a given abstraction cannot be pre-determined.

Example 1.1 Consider, e.g., the following β-reduction divergence [13]:

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

One branch requires the renaming of an x into, say, a z and, although the resulting terms are α-equivalent, they are not equal as first-order abstract syntax.

1.1 Side-Stepping the Problem

Standard informal practice in pen-and-paper proofs is simply to ignore variable names as epitomised in the 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 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."

In a formal setting, postfixed name-unification could be used to prove an equational property in the above example but the status of the obtained result will need some interpretation. Furthermore, the required technology is not easily managed in a theorem prover (see, for example, [23] where this has been accomplished for β-confluence by building on top of a proof for a de Bruijn language).

1.2 Formal Alternatives

In response to the long-since established and well-recognised difficulties with names and equational proofs over first-order abstract syntax with one-sorted variable names [7,13,22], a large number of alternative formalisms have been

[5] Thanks to L. Regnier for pointing out that parallel substitutions can overcome this and to an anonymous reviewer for referring us to [24] where the details are given.

proposed [7,8,9,10,11,12,14,20]. The alternatives are motivated by formalist considerations and, generally speaking, seek to overcome the naming problems by native means. All the proposals mark a conceptual and formal departure from the naive qualities of the first-order set-up.

1.3 Consolidating the Established Pen-and-Paper Practices

We wish to argue that the informal pen-and-paper proof practices of the wider programming language theory community are formalisable — and feasibly so. The point is that while the informal proof practices certainly are not mechanisable in general, they can indeed be employed under suitable initial conditions. In particular, we show that the key lemmas of various equational properties typically can be established up-to an initiality condition which is essentially a formal variant of the BVC. Technically speaking, the proofs of the lemmas we consider here only require reduction of *residuals* (i.e., descendants of terms under reduction) to go through [4]. Informally, this means that the reductions being performed could have been made in the original term. By the initiality condition we use, variable conflicts between residuals of different subterms are not possible. Neither are they possible between residuals of the same subterm by Hyland's Disjointness Property, which simply says that such residuals are non-overlapping (if not, one would be inside the other and a newly created redex would have had to have been contracted) [18].

1.4 The Remaining Proof Burden

Having established an appropriate conditional variant of the key lemma for an equational property, it remains to be seen that the lemma which is actually needed for the abstract reasoning to proceed can be obtained. To this end, we employ diagrammatic reasoning to compose commutativity lemmas for the involved relations. Although the proofs of the individual lemmas are fairly detailed (as formal proofs are), their statement as commutative diagrams and their subsequent composition are easily understood by a reader.[6]

The insights into equational properties that are communicated succinctly by the use of the 'right' definitions and abstract rewriting technology are thus not compromised in any way by our method [6,16,17,25]. The only intuitive novelty is an administrative proof layer, dealing with naming issues. It serves to bridge and facilitate the formalisation of the two standard pen-and-paper practices (as listed at the start of the paragraph).

1.5 Context of Work

This work builds on Wells and the first author's [28] and the authors' [27].

[6] On a personal note, we also find that the methodology is helpful in analysing and decomposing a sought-after proof.

A Calculus of Linking

The basic techniques we use were pioneered mainly by Vestergaard for [28]. In there, we prove the strong finite development property (SFDP), cf. Section 6, for a calculus of linking with first-class primitive modules: the m-calculus. The m-calculus is noticeable from our current perspective in allowing mutually recursive binding amongst collections of elements. In our opinion, freely assuming the relevant notion of α-equivalence plus AC-equivalence over an inductively-defined list-representation of sets in the m-calculus would be questionable, even at the informal level.

Mechanisation

The proof method at hand is, among other things, substantiated as being amenable to mechanisation in [27] where we account for a mechanised β-confluence proof. The mechanisation specifics of the proof development are accounted for in the second author's Honours dissertation [2].

Other Uses

The proof methodology has also been applied by the first author to prove $\beta\eta$-confluence, η-over-β postponement, and β-standardisation [26]. The first of these follows the pattern we account for here in being concerned with (horizontal) commutativity whereas the latter two are radically different in being concerned with distributivity, or vertical commutativity, of relations.[7]

1.6 This Paper

Section 2 presents the basics of first-order representations of higher-order languages. In Section 3, we establish that the residual theory of β in the λ-calculus is renaming-free up-to an initiality condition. This means that a host of key lemmas for equational properties (of the λ-calculus) can be established in suitably restricted forms by primitive means, only. Section 4 shows how to use this to prove the equational part of the SFDP. In Section 5, we show that the established property implies confluence in a non-standard way (i.e., without finiteness of developments). In Section 6, we account for two other uses of the proof method [27,28]. Finally, in Section 7, we conclude.

2 Raw and Real Calculi

In order to reason about, say, the λ-calculus proper in a first-order manner, we enforce a distinction between *raw* and *real* calculi. The former are inductively defined structures (and we use dashed arrows to denote their reduction relations). The latter are obtained from the former by collapsing under an

[7] By (horizontal) commutativity we mean resolution of co-initial divergences. By distributivity, or vertical commutativity, we mean reversal of orders of chained relations.

equivalence relation (and their relations are denoted by full-lined arrows). We detail the distinction in the following definition pertaining to Abstract Rewrite Systems (ARS) — an ARS is a collection of binary relations on the same domain.

Definition 2.1 Consider an ARS of the form: $\dashrightarrow_s, \dashrightarrow_c \subseteq A \times A$ (the *raw* calculus). Its *structural collapse* is induced from $\dashrightarrow_c$ by factoring out the equivalence relation generated by $\dashrightarrow_s$; the new, *real*, relation is written $\longrightarrow_c$:

- $A^s \;=\; A/\!=\!=_s$
- $\lfloor - \rfloor_s : A \longrightarrow A^s$
  $\qquad M \mapsto \{N \mid M =\!=_s N\}$
- $\lfloor M \rfloor_s \longrightarrow_c \lfloor N \rfloor_s \Leftrightarrow M =\!=_s ; \dashrightarrow_c ; =\!=_s N$

To justify the construction, we present the following two results, essentially from [27]. Please consult Appendix A for details of our diagram notation. The first is a minimal requirement: the raw and real equational theories coincide.

Proposition 2.2 *Consider an ARS* $\dashrightarrow_s, \dashrightarrow_c \subseteq A \times A$ *and its structural collapse.*

$$A/\!=\!=_{s \cup c} = A^s/\!=\!=_c$$

The second gives a sufficient criterion for the raw/real equivalence of not only equational theories but of the ultimate equational property: confluence.

Lemma 2.3 *Consider an ARS* $\dashrightarrow_s, \dashrightarrow_c \subseteq A \times A$ *and its structural collapse.*[8]

$$\Rightarrow \quad \left(\begin{array}{l} \lfloor M \rfloor_s \longrightarrow\!\!\!\!\rightarrow_c \lfloor N \rfloor_s \;\Leftrightarrow\; M \dashrightarrow\!\!\!\!\rightarrow_{s \cup c} N \\ \wedge\ \mathrm{Confl}(\longrightarrow_c) \;\Leftrightarrow\; \mathrm{Confl}(\dashrightarrow_{s \cup c}) \end{array} \right)$$

The confluence equivalence result is perhaps not too surprising as presented but we refer the interested reader to [27] for a comprehensive account of its negative variants and their implications. For example, standard pen-and-paper confluence proofs [1] can easily be established as incomplete.

3 Barendregt-Style Reasoning is Correct, Sometimes

In order to suggest the general applicability of our proof methodology, we will now prove that the residual theory of β-reduction in the (raw) λ-calculus is renaming-free up-to an initiality condition. The implication is that all equational reasoning about the relevant fragment of the residual theory of β can be undertaken with primitive proof principles. Residual theory is very low level from a proof-technology point-of-view and, as we shall see, often need not be considered independently when doing equational proofs. Residual theory

[8] Double-headed arrows are transitive, reflexive relations. Confl is confluence.

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

$$
\begin{aligned}
\mathrm{FV}(y) &= \{y\} \\
\mathrm{FV}(e_1 \star e_2) &= \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) \\
\mathrm{FV}(\lambda y.e) &= \mathrm{FV}(e) \setminus \{y\}
\end{aligned}
$$

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

Fig. 1. Total but partially correct substitution, $-[- := -]$, free variables, $\mathrm{FV}(-)$, and variables capturing free occurrences of x, $\mathrm{Capt}_x(-)$, for $\Lambda^{\mathrm{var}}_{\mathrm{res}}$. NB! The infix operator $\_\star\_$ is used to range over marked and ordinary application: $\_@\_$ and $\_\ \_$.

is relevant as the key lemmas of most equational properties respect residual theory in the sense that their proofs only require the contraction of residuals.

The results in this section have all been mechanised in Isabelle/HOL and the proof scripts are available from our homepages. The mechanisation took roughly a week for one person.

Definition 3.1 [The Raw, Marked λ-Calculus] Terms are given by

$$\Lambda^{\mathrm{var}}_{\mathrm{res}} ::= x \ \mid\ \Lambda^{\mathrm{var}}_{\mathrm{res}}\Lambda^{\mathrm{var}}_{\mathrm{res}} \ \mid\ \lambda x.\Lambda^{\mathrm{var}}_{\mathrm{res}} \ \mid\ (\lambda x.\Lambda^{\mathrm{var}}_{\mathrm{res}}) \mathbin{@} \Lambda^{\mathrm{var}}_{\mathrm{res}}$$

The *residual* relations are given in Figure 2 with auxiliaries in Figure 1.

Only *marked redexes* (denoted by "@") can be contracted. The marks

$$\frac{\mathrm{FV}(e_2) \cap \mathrm{Capt}_x(e_1) = \emptyset}{(\lambda x.e_1) \mathbin{@} e_2 \dashrightarrow_{\beta^{\mathrm{res}}} e_1[x := e_2]}\,(\beta^{\mathrm{res}})$$

$$\frac{e_1 \dashrightarrow_{\beta^{\mathrm{res}}} e_1'}{e_1 \star e_2 \dashrightarrow_{\beta^{\mathrm{res}}} e_1' \star e_2}\,(\mathrm{A1}_{\beta^{\mathrm{res}}}) \qquad \frac{e_2 \dashrightarrow_{\beta^{\mathrm{res}}} e_2'}{e_1 \star e_2 \dashrightarrow_{\beta^{\mathrm{res}}} e_1 \star e_2'}\,(\mathrm{A2}_{\beta^{\mathrm{res}}})$$

$$\frac{e \dashrightarrow_{\beta^{\mathrm{res}}} e'}{\lambda x.e \dashrightarrow_{\beta^{\mathrm{res}}} \lambda x.e'}\,(\mathrm{L}_{\beta^{\mathrm{res}}})$$

$$\frac{e_1 \dashrightarrow_{\beta^{\mathrm{res}}} e_1' \quad e_2 \dashrightarrow_{\beta^{\mathrm{res}}} e_2' \quad \mathrm{FV}(e_2') \cap \mathrm{Capt}_x(e_1') = \emptyset \quad x \in \mathrm{FV}(e_1')}{(\lambda x.e_1) \mathbin{@} e_2 \dashrightarrow_{\beta^{\mathrm{res}}} e_1'[x := e_2']}\,(\beta^{\mathrm{res}*})$$

$$\frac{e_1 \dashrightarrow_{\beta^{\mathrm{res}}} e_1' \qquad x \notin \mathrm{FV}(e_1')}{(\lambda x.e_1) \mathbin{@} e_2 \dashrightarrow_{\beta^{\mathrm{res}}} e_1'}\,(\beta^{\mathrm{res}*}_{\mathrm{lazy}})$$

$$\frac{}{x \dashrightarrow_{\beta^{\mathrm{res}}} x}\,(\mathrm{Var}^{\beta^{\mathrm{res}}}_{*}) \qquad \frac{e \dashrightarrow_{\beta^{\mathrm{res}}} e'}{\lambda x.e \dashrightarrow_{\beta^{\mathrm{res}}} \lambda x.e'}\,(\mathrm{L}^{\beta^{\mathrm{res}}}_{*})$$

$$\frac{e_1 \dashrightarrow_{\beta^{\mathrm{res}}} e_1' \qquad e_2 \dashrightarrow_{\beta^{\mathrm{res}}} e_2'}{e_1 e_2 \dashrightarrow_{\beta^{\mathrm{res}}} e_1' e_2'}\,(\mathrm{A}^{\beta^{\mathrm{res}}}_{*})$$

Fig. 2. Raw residual relations for the marked λ-calculus: one-step reduction and residual-completion, respectively.

are introduced to prevent newly created redexes from being contracted. The disjointness of *free and capturing variables* is the smallest predicate ensuring that no (raw) β-reduction will result in a variable clash. [9] The relations we use are respectively the one-step residual and residual-completion relations. The latter contracts all marked redexes (if possible) without leaving any behind. It includes the lazy β-rule for technical reasons (without it, Lemma 4.8 requires an initiality condition).

[9] In [27], we expound the fact that a raw version of the λ-calculus defined in accordance with the style used here structurally collapses to the standard real λ-calculus [5,13].

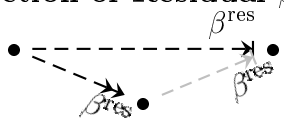
Definition 3.2 [Barendregt Conventional Form] A term is said to be a BCF if each binder name is unique and different from the free variables in the term.

Lemma 3.3 (BCF**-Enabling of Raw Residual-Completion)**

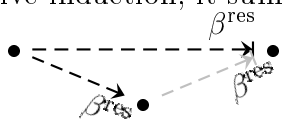
(BCF) 

Proof. By structural induction in the initial BCF. The only non-trivial case is for $(\lambda x.e_1)\,@\,e_2$. It follows by *monotonicity* of bound and free (but not capturing) variables under $\dashrightarrow_{\beta^{\mathrm{res}}}$. The BCF-condition ensures that no changes to e_1 can block the redex as capturing variables are included in bound variables.□

Lemma 3.4 (Raw Completion of Residual β)



Proof. By transitive, reflexive induction, it suffices to prove:



The proof is by rule induction in $\dashrightarrow_{\beta^{\mathrm{res}}}$. The interesting cases are (β^{res}) and $(\beta^{\mathrm{res}}_{\mathrm{lazy}})$. The proof follows from Substitutivity and Substitution Lemmas. □

Substitutivity and Substitution Lemmas are non-trivial to prove formally. For our present purposes we will merely present one of each to give an indication of the style. For the following lemmas, we note 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 3.5 (Marked Substitution)

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

Lemma 3.6 (β Residual-Completion Substitutivity)

$$e_1 \dashrightarrow_{\beta^{\mathrm{res}}} e_1' \;\wedge\; e_2 \dashrightarrow_{\beta^{\mathrm{res}}} e_2'$$
$$\wedge\, (\mathrm{Capt}_x(e_1) \cap \mathrm{FV}(e_2) = \emptyset) \;\wedge\; (\mathrm{Capt}_x(e_1') \cap \mathrm{FV}(e_2') = \emptyset)$$
$$\Downarrow$$
$$e_1[x := e_2] \dashrightarrow_{\beta^{\mathrm{res}}} e_1'[x := e_2']$$

We refer the interested reader to the Isabelle/HOL proof development at our homepages for full details. The main result of this section is the following.

Theorem 3.7 (BCF**-Initial Raw β-Residual Theory is not Blocked)**

Proof. By Lemma 3.3, any BCF residual-completes. By Lemma 3.4, such a completion can absorb any initial $\dashrightarrow\!\!\!\!\twoheadrightarrow_{\beta^{\mathrm{res}}}$. □

The theorem states that any residual of a BCF is such that any marked redex in it (except, possibly, for those that are discarded by the lazy β-rule) can be contracted without resulting in a variable clash. We believe the lazy β-rule could be avoided for the purposes of the above result. The conclusion would then be that *all* marked redexes can be contracted. However, this would greatly complicate the proof of Lemma 3.4 as the transitive, reflexive induction no longer would be straightforward. The reason is that the considered property would need to be formulated with, say, BCF-initiality which is not preserved along the inducted relation. As it stands, the only enforced restriction is that $\dashrightarrow\!\!\!\!\!\rightarrow_{\beta^{\mathrm{res}}}$ must be well-defined in any non-trivial cases (of which there are some, cf. Lemma 3.3).

The results of this section are also substantial in a more traditional analysis as we show next.

4 Strong (Finite) Development and Consequences

In this section, we will lift the (raw) results of the previous section to the real level. The resulting property comprises the major part of the Strong Finite Development Property (SFDP) for the λ-calculus (see, e.g., [1, Theorem 11.2.25]).

Definition 4.1 Let $\rightarrow$ be a residual relation, i.e., a relation which only contracts marked redexes (such as $\dashrightarrow_{\beta^{\mathrm{res}}}$ in Section 3).

- $\twoheadrightarrow$ is said to be the corresponding *development relation*
- $e \twoheadrightarrow e'$ is said to be a *development* (of e)
- a development, $e \twoheadrightarrow e'$, is said to be *complete* if unMarked(e'). [10]

Definition 4.2 [The SFDP] A relation, $\rightarrow$, is said to enjoy the *strong finite development property* if

(i) SN($\rightarrow$)
— developments are finite

[10] With the obvious definition of unMarked($-$).

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

$$\frac{e_1 \overset{y}{\dashrightarrow}_{i\alpha^{\mathrm{res}}} e_1'}{e_1 \star e_2 \overset{y}{\dashrightarrow}_{i\alpha^{\mathrm{res}}} e_1' \star e_2}\,(\mathrm{A1}_{\alpha^{\mathrm{res}}}) \qquad \frac{e_2 \overset{y}{\dashrightarrow}_{i\alpha^{\mathrm{res}}} e_2'}{e_1 \star e_2 \overset{y}{\dashrightarrow}_{i\alpha^{\mathrm{res}}} e_1 \star e_2'}\,(\mathrm{A2}_{\alpha^{\mathrm{res}}})$$

Fig. 3. The raw indexed α^{res}-relation.

(ii) $e \twoheadrightarrow e' \Rightarrow (\exists e''.e' \twoheadrightarrow e'' \wedge \mathrm{unMarked}(e''))$ [11]
— developments can be completed

(iii) $(e \twoheadrightarrow e_1 \wedge e \twoheadrightarrow e_2 \wedge \mathrm{unMarked}(e_i)) \Rightarrow e_1 = e_2$
— completions are unique

Intuitively, developments are finite because newly constructed redexes are not marked and thus cannot be contracted.

We call the result we prove the S(F)DP in that it excludes the finiteness of developments. In contrast to conventional wisdom, [12] we show that even without finiteness of developments, the SFDP (that is, the S(F)DP) can be used to conclude confluence. The proof method is a hybrid of the standard SFDP method and the Tait/Martin-Löf/Takahashi method which we review in Section 6. It seems to be combinatorially less complex than both of these because it relies on a 'weaker' β-commutativity lemma. On the other hand, it might also be a bit longer because it uses more auxiliary lemmas. The proof appears to be new.

We refer the interested reader to [15] for a comprehensive, mechanised account of the residual theory of the λ-calculus represented with de Bruijn terms. The aim of [15] is a theoretically robust and stand-alone treatment of residual theory which is essentially categorical. Informally, Huet's results are to ours as the original Tait/Martin-Löf confluence proof using parallel reduction is to Takahashi [25], cf. Sections 5 and 6.2.

4.1 Structural/Computational Commutativity

A central part of our proposed proof methodology is the use of commutative diagrams to reduce general equational properties to conditional variants that can be proved by simple inductive methods. The main notion of commutativity we use is between computational relations proper (e.g., β^{res}) and relations

[11] Observe that $e' \twoheadrightarrow e''$ only can contract residuals of e as the terms are marked.

[12] See, e.g., [1, p.283]: "[The finiteness of developments] has important consequences, among them being [confluence] ..."

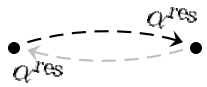
which axiomatise structural equivalence. In the raw, marked λ-calculus, the relevant notion of structure is the α^{res}-relation, cf. Figure 3 (which like most other formal statements in this section, are adapted from [27]). The figure presents an indexed relation which is intended to smoothly facilitate the following definition.

Definition 4.3 Following Figure 3,

- let the α^{res}-relation be: $e \dashrightarrow_{\alpha^{\mathrm{res}}} e' \Leftrightarrow \exists y.e \overset{y}{\dashrightarrow}_{i\alpha^{\mathrm{res}}} e'$
- let fresh-naming α_0^{res} be: $e \dashrightarrow_{\alpha_0^{\mathrm{res}}} e' \Leftrightarrow \exists y.e \overset{y}{\dashrightarrow}_{i\alpha^{\mathrm{res}}} e' \wedge y \notin \mathrm{Var}(e)$ [13]

We stress that the α^{res}-equivalence relation induced from the above α^{res}-relation is the standard notion (up-to the marks). With reference to Lemma 2.3, we then present the following result.

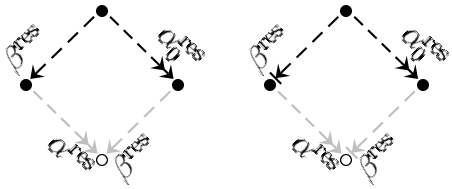
Lemma 4.4 ($\dashrightarrow_{\alpha^{\mathrm{res}}}$-Symmetry)



Proof. By rule induction in the underlying indexed relation $\dashrightarrow_{i\alpha^{\mathrm{res}}}$. The only non-trivial case follows by basic 'renaming sanity' properties of substitution.□

The reason for presenting two α-relations is that a fully general commutativity result for α^{res} and β^{res} is not to be expected as there are side-conditions pertaining to variable names involved. On the other hand, it *is* to be expected that no α_0^{res}-step will block a β^{res}-step.

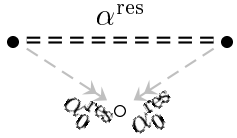
Lemma 4.5



Proof. By transitive, reflexive induction in $\dashrightarrow\!\!\!\!\!\rightarrow_{\alpha_0^{\mathrm{res}}}$. The reflexive and transitive cases are trivial. In the base case we proceed by rule induction in the underlying indexed relation: $\overline{\dashrightarrow}_{i\alpha_0^{\mathrm{res}}}$. The proof is rather involved and uses the relevant Substitutivity lemmas on $\overline{\dashrightarrow}_{i\alpha^{\mathrm{res}}}$ as well as a Substitution Lemma. We also take advantage of the fact that it suffices to use the same name for each step of the $\overline{\dashrightarrow\!\!\!\!\!\rightarrow}_{i\alpha^{\mathrm{res}}}$-closure. □

It is not difficult to see that $\dashrightarrow_{\alpha_0^{\mathrm{res}}} \subset \dashrightarrow_{\alpha^{\mathrm{res}}}$. It can also be shown that $\dashrightarrow\!\!\!\!\!\rightarrow_{\alpha_0^{\mathrm{res}}} \subset \dashrightarrow\!\!\!\!\!\rightarrow_{\alpha^{\mathrm{res}}}$. However, by the following lemma we have $==_{\alpha^{\mathrm{res}}} = ==_{\alpha_0^{\mathrm{res}}}$.

Lemma 4.6



[13] With $\mathrm{Var}(e)$ being the variable names occurring (anywhere) in e.

Proof. Intuitively the proof is straightforward — since the structure of both terms is the same, it suffices to rename the two α-equivalent terms with sufficiently many fresh names in the same order in order to arrive at the same term. Formally, the proof proceeds by transitive, reflexive induction (by Lemma 4.4, it suffices to induct over $\dashrightarrow\!\!\!\!\!\rightarrow_{\alpha^{\mathrm{res}}}$) and then by rule induction in $\dashrightarrow_{\alpha^{\mathrm{res}}}$. The formal proof is complicated greatly by the need for explicit quantification over lists of variable names used for the reduction sequences. Substitutivity and Substitution Lemmas are also required. The proofs do not provide any insights and are omitted here; for a fuller exposition the reader is invited to consult [27]. □

The reason for the above asymmetry between α^{res} and α_0^{res} is that not all terms can be the target (although they naturally can be the source) of a $\dashrightarrow_{\alpha_0^{\mathrm{res}}}$-step, whereas some terms are not too difficult to target:

Lemma 4.7

$$\bullet \overset{\alpha_0^{\mathrm{res}}}{\dashrightarrow\!\!\!\!\!\rightarrow} \circ \,(\mathrm{BCF})$$

Proof. Intuitively, the property holds since renaming every λ-binder occurring in the term with a fresh name will yield a BCF term. To prove the result formally, we first prove the following property by structural induction on e_1:

$$\forall\, \vec{z_i}, e_1.\ ||\, \vec{z_i}\, || = \#_\lambda(e_1) \wedge \{z_i\} \text{ all different} \wedge (\{z_i\} \cap (\mathrm{FV}(e_1) \cup \mathrm{BV}(e_1)) = \emptyset)$$

$$\Downarrow$$

$$\exists e_2 . e_1 \overset{\vec{z_i}}{\dashrightarrow\!\!\!\!\!\rightarrow}_{i\alpha_0} e_2 \wedge \mathrm{BCF}(e_2) \wedge \{z_i\} = \mathrm{BV}(e_2)$$

where $\#_\lambda(e_1)$ is the number of λ-abstractions in e_1. The inclusion of the variable name information is necessary in order to make the proof fully constructive.

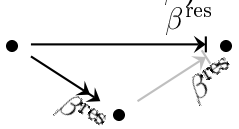
□

With the relevant raw commutativity lemmas in place, we are now ready to prove properties at the real level.

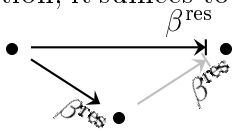
4.2 The S(F)DP for the Real λ-Calculus

With a danger of causing confusion, we retain e as a meta-variable over real as opposed to raw terms (and occasionally use M and N for raw terms instead). We also use, e.g., the raw notion of unMarked$(-)$ at the real level. On the other hand, we strictly distinguish raw and real relations by the use of dashed respectively full-coloured arrows. This convention disambiguates our notation.

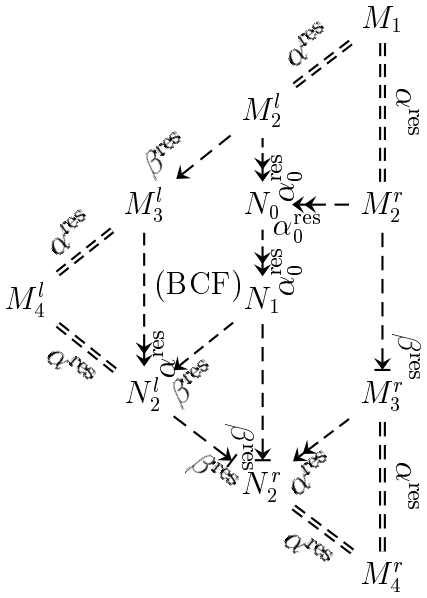
Lemma 4.8 (Real Completion of Residual β)



Proof. By transitive, reflexive induction, it suffices to prove the one-step case.

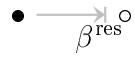


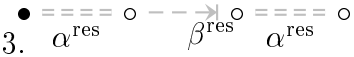
The following diagram outlines the proof of the definitional re-statement of the property at the raw level.



For the Ms given, we can construct the Ns in the divergence resolution on the left in order (0, 1, ...). N_0 follows from Lemma 4.6. N_1 from Lemma 4.7. The N_2's from Lemma 4.5. The lower leg follows by transitivity of $==_{\alpha^{\text{res}}}$ and Lemma 3.4.

□

Lemma 4.9 (Real Residual-Completion) 

Proof. By Definition 2.1, we must prove: 
The result follows from Lemmas 4.7 and 3.3. □

Proposition 4.10 (Further Properties of Real Residual-Completion)

(i) $\twoheadrightarrow_{\beta^{\text{res}}}$ *is functional*

(ii) $e \twoheadrightarrow_{\beta^{\text{res}}} e' \Rightarrow \text{unMarked}(e')$ — *residual-completion leaves no marks*

(iii) $\twoheadrightarrow_{\beta^{\text{res}}} \subseteq \longrightarrow\!\!\!\!\!\!\twoheadrightarrow_{\beta^{\text{res}}}$ — *residual-completion is a development*

Proof. It essentially suffices to establish the properties at the raw level.

(i) The generating rules of the relation are mutually disjoint.

(ii) And, they contract all marked redexes without creating new ones.

(iii) By inspection (i.e., rule induction).

□

Theorem 4.11 $\longrightarrow_{\beta^{\text{res}}}$ *enjoys the S(F)DP.*

Proof. Developments can be completed (i) as any term residual-completes by Lemma 4.9, (ii) as residual-completions are complete developments by Propo-

sition 4.10, and (iii) as developments are transitive by definition. The resulting term is unique by functionality of $\twoheadrightarrow_{\beta^{\mathrm{res}}}$ (Proposition 4.10) and Lemma 4.8.□

We observe that an implication of the fact that developments complete is that the residual relation is weakly normalising.

5 Confluence from the S(F)DP

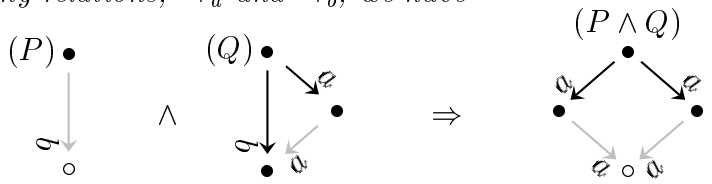
The original justification for considering residuals is the fact that developments (as opposed to unrectricted β-reduction) can be proved finite [4,22]. By using Newman's Lemma,[14] the development relation can thus be proved confluent merely by proving it weakly confluent. In turn, this can be used to prove the λ-calculus proper confluent, as we will detail in Section 5.2.

As already stated, conventional wisdom has it that it is the finiteness of developments which is the main cause of confluence when considering residuals.[15] In contrast to this, we will now show how to obtain confluence from the S(F)DP (which requires weak but not strong normalisation of the residual relation). We need a few simple ARS results (known, e.g., from the Tait/Martin-Löf/Takahashi method for proving confluence, cf. Section 6).

Lemma 5.1 $(\exists \to_b \,.\, \to_a \subseteq \to_b \subseteq \twoheadrightarrow_a \;\wedge\; \diamond(\to_b)) \;\Rightarrow\; \mathrm{Confl}(\to_a)$

Proof. A formalisation is provided in [21] and is re-used in [27]. □

Lemma 5.2 (Guarded Diamond Diagonalisation) *For any predicates, P and Q, and any relations, $\to_a$ and $\to_b$, we have*



Proof. Straightforward. A formalisation is provided in [27] following [25]. □

5.1 Residual Confluence

We believe the proof of the following result is new. We let it speak for itself.

Lemma 5.3 $\mathrm{Confl}(\longrightarrow_{\beta^{\mathrm{res}}}) \;\wedge\; \mathrm{Confl}(\dashrightarrow_{\alpha^{\mathrm{res}} \cup \beta^{\mathrm{res}}})$

Proof. As for the first result, we can see that the triangle-property premise of Lemma 5.2 is substantiated by Lemma 4.8 as any residual-completion is a development (Proposition 4.10). The other premise of Lemma 5.2 is given

[14] Weak confluence and strong normalisation implies confluence.

[15] See, e.g., [1, p.283]: "[The finiteness of developments] has important consequences, among them being [confluence] ..."

by Lemma 4.9. The second result follows from the first and Lemma 4.4 by Lemma 2.3. □

5.2 Confluence Proper

This section uses standard techniques and is provided virtually uncommented.

Definition 5.4 Let $|-|: \Lambda^{\mathrm{var}}_{\mathrm{res}} \longrightarrow \Lambda^{\mathrm{var}}$ be the function that removes marks from a marked raw term (with the obvious definition of Λ^{var}). Let

$$|\, e_1 \,| \longrightarrow_{\mathrm{dev}} |\, e_2 \,| \;\Leftrightarrow^{\mathrm{def}}\; e_1 \longrightarrow\!\!\!\!\!\rightarrow_{\beta^{\mathrm{res}}} e_2$$

be the *non-marked development relation.*

Proposition 5.5 $\longrightarrow_\beta \subseteq \longrightarrow_{\mathrm{dev}} \subseteq \longrightarrow\!\!\!\!\!\rightarrow_\beta$

Proof. Both inclusions follow by simole rule inductions at the raw level. □

Theorem 5.6 $\mathrm{Confl}(\longrightarrow_\beta) \;\wedge\; \mathrm{Confl}(\dashrightarrow_{\alpha\cup\beta})$

Proof. As for the first result, the $\longrightarrow_{\mathrm{dev}}$-relation is a witness of the quantification in Lemma 5.1 according to Lemma 5.3 and Proposition 5.5. The second result follows from the first and an adaptation of Lemma 4.4 by Lemma 2.3.□

6 Other First-Order Equational Proofs

In this section we give an overview of the diagrammatic reasoning of [27,28]. The (formats of the necessary) commutativity lemmas have already been presented. We stress that [28] introduces the m-calculus, a calculus of linking with first-class primitive modules. The m-calculus is vastly more expressive than the λ-calculus in terms of the computational paradigms in can simulate *directly* [28]. Among other things, it contains mutually recursive binding amongst sets of elements. For uniformity, we frame both results in terms of the λ-calculus.

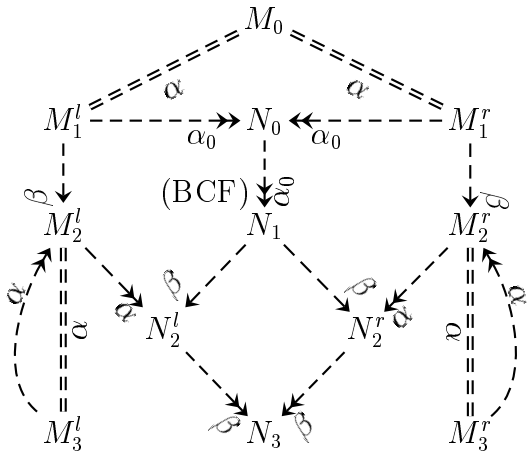
6.1 The Strong Finite Development Property [28]

Apart from finiteness of developments, it suffices to prove a real residual relation weakly confluent to prove the SFDP and confluence. The latter step uses Newman's Lemma. The former is conducted at the raw level.

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

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

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

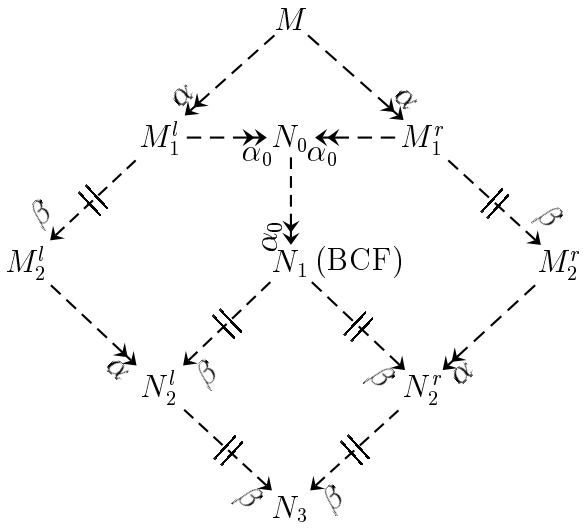


The M-divergence on the left is the re-statement at the raw level of a one-step divergence in the real calculus according to Definition 2.1. We have suppressed the "res" superscripts on the residual-relation names. The N's are constructed in order $(0, 1, \ldots)$ to resolve the divergence. Adaptations of Lemmas 4.4, 4.5, 4.6, and 4.7 are easily recognised in the proof.

6.2 Confluence via Parallel Reduction [27]

The Tait/Martin-Löf proof method uses a parallel relation that can contract any number of pre-existing redexes in one step, -H→ below. The point is that such a relation can be shown to be a witness for the quantification in Lemma 5.1. Takahashi [25] introduced the idea of defining the parallel relation directly by induction on terms which brings the method within reach of our proof methodology. Figure 4 presents our raw version of the relation [27].

Takahashi also introduced the idea of diagonalising the central diamond property of the proof resolution, cf. Lemma 5.2, which we re-use in [27]. We do not present the details here but refer to Lemma 5.3.



The proof resolution technique on the left is assumed to be familiar to the reader at this point. Instead, we note that the proof establishes $\diamond(\dashrightarrow\!\!\!\!\!\!\rightarrow_\alpha; \dashrightarrow\!\!\!\!\!\!\rightarrow_\beta)$ — the symbol ";" is for relation composition. As opposed to the other two proofs we have presented, we use this property to prove (raw) $\mathrm{Confl}(\dashrightarrow_{\alpha\cup\beta})$ first (Lemma 5.1) and from there conclude (real) $\mathrm{Confl}(\longrightarrow_\beta)$ via Lemma 2.3.

7 Conclusion

We hope to have convinced our reader that the pen-and-paper proof practices of the wider programming language theory community are formalisable. In trying to do so, we have proved an approximation of the SFDP of the λ-calculus. It was observed that the presented results imply that the residual theory of β-reduction is renaming-free up-to BCF-initiality. The results also imply confluence. Although we use the development set-up, we showed that strong normalisation of the residual relation is not required to draw this conclusion. Following on from there, we accounted for our work in [27,28] and outlined the administrative proof layer that is required to formalise the SFDP and confluence for a higher-order language represented with first-order abstract syntax over one-sorted variable names. On-going work aims at extending the methodology to other typical situations. Although informal proofs are incomplete by a wide formal margin as far as proof burden resolution goes, we hope to also have convinced our reader that the intuitive gap is somewhat smaller.

References

[1] Barendregt, H., "The Lambda Calculus — Its Syntax and Semantics (Revised Edition)," North-Holland, 1984.

[2] Brotherston, J., "Formalizing Proofs in Isabelle/HOL of Equational Properties for the Lambda-Calculus using One-Sorted Variable Names", Honours dissertation, University of Edinburgh (2001). Available from the author's homepage.

[3] Burstall, R., *Proving properties of programs by structural induction*, The Computer Journal **12** (1967).

[4] Church, A. and J. B. Rosser, *Some properties of conversion*, Transaction of the American Mathematical Society **39** (1936).

[5] Curry, H. B. and R. Feys, "Combinatory Logic," North-Holland, Amsterdam, 1958.

[6] David, R., *Une preuve simple de rèsultats classiques en λ calcul*, Comptes Rendus de l'Acadèmie des Sciences **320** (1995), pp. 1401–1406, série I.

[7] de Bruijn, N., *Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser Theorem*, Indag. Math. **34** (1972), pp. 381–392.

[8] Despeyroux, J. and A. Hirschowitz, *Higher-order abstract syntax with induction in Coq*, in: F. Pfenning, editor, *Proceedings of LPAR-5*, LNAI **822** (1994).

[9] Despeyroux, J., F. Pfenning and C. Schürmann, *Primitive recursion for higher-order abstract syntax*, in: P. De Groote and J. R. Hindley, editors, *Proceedings of TLCA-3*, LNCS **1210** (1997).

[10] Fiore, M., G. Plotkin and D. Turi, *Abstract syntax and variable binding*, in: Longo [19], pp. 193–202.

[11] Gabbay, M. J. and A. M. Pitts, *A new approach to abstract syntax involving binders*, in: Longo [19], pp. 214–224.

[12] Gordon, A. D. and T. Melham, *Five axioms of alpha-conversion*, in: J. Von Wright, J. Grundy and J. Harrison, editors, *Proceedings of TPHOL-9*, LNCS **1125** (1996).

[13] Hindley, J. R., "The Church-Rosser Property and a Result in Combinatory Logic," Ph.D. thesis, University of Newcastle upon Tyne (1964).

[14] Hofmann, M., *Semantical analysis of higher-order abstract syntax*, in: Longo [19], pp. 204–213.

[15] Huet, G., *Residual theory in λ-calculus: A formal development*, Journal of Functional Programming **4** (1994), pp. 371–394.

[16] Joachimski, F. and R. Matthes, *Standardization and confluence for a lambda calculus with generalized applications*, in: L. Bachmair, editor, *Proceedings of RTA-11*, LNCS **1833** (2000).

[17] Joachimski, F. and R. Matthes, *Short proofs of normalization for the simply-typed lambda-calculus, permutative conversions and Gödel's T*, Archive for Mathematical Logic (200X), to appear.

[18] Klop, J. W., "Combinatory Reduction Systems," Mathematical Centre Tracts 127, Mathematisch Centrum, Amsterdam, 1980.

[19] Longo, G., editor, "Proceedings of LICS-14," IEEE CS Press, 1999.

[20] McKinna, J. and R. Pollack, *Some lambda calculus and type theory formalized*, Journal of Automated Reasoning **23** (1999).

[21] Nipkow, T., *More Church-Rosser proofs (in Isabelle/HOL)*, in: *Proceedings of CADE-13*, LNCS **1104** (1996).

[22] Schroer, D. E., "The Church-Rosser theorem," Ph.D. thesis, Cornell (1965).

[23] Shankar, N., *A mechanical proof of the Church-Rosser Theorem*, Journal of the ACM **35** (1988), pp. 475–522.

[24] Stoughton, A., *Substitution revisited*, Theoretical Computer Science **59** (1988), pp. 317–325.

[25] Takahashi, M., *Parallel reductions in λ-calculus*, Information and Computation **118** (1995), pp. 120–127.

[26] Vestergaard, R., "First-Order Equational Reasoning about Higher-Order Languages," Ph.D. thesis, Heriot-Watt University (2001), forthcoming.

[27] Vestergaard, R. and J. Brotherston, *A formalised first-order confluence proof for the λ-calculus using one-sorted variable names (*Barendregt was right after all ... almost*)*, in: A. Middeldorp, editor, *Proceedings of RTA-12*, LNCS **2051** (2001).

[28] Wells, J. and R. Vestergaard, *Equational reasoning for linking with first-class primitive modules*, in: G. Smolka, editor, *Proceedings of ESOP-9*, LNCS **1782** (2000).

A Commutative Diagrams

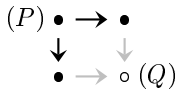
Formally, a commutative diagram is a set of vertices and a set of directed edges between pairs of vertices. A vertex is written as either $\bullet$ or $\circ$. Informally, this denotes quantification modes over terms, universal respectively existential. A vertex may be guarded by a predicate. Edges are 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:

(i) write universal quantifications for all $\bullet$s (over the relevant domains)

(ii) assume the full-coloured relations and the validation of any guard for a $\bullet$

(iii) conclude the guarded existence of all $\circ$s and their relations

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



$$\forall e_1, e_2, e_3 \in A\,.\;\; e_1 \to e_2 \;\wedge\; e_1 \to e_3 \;\wedge\; P(e_1)$$
$$\Downarrow$$
$$\exists e_4 \in A\,.\, e_2 \to e_4 \;\wedge\; e_3 \to e_4 \;\wedge\; Q(e_4)$$

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