*Electronic Notes in Theoretical Computer Science 58 No. 1 (2001)*

*URL:* [*http://www.elsevier.nl/locate/entcs/volume58.htm*](http://www.elsevier.nl/locate/entcs/volume58.html)*l 19 pages*

*The Mechanisation of* Barendregt-Style Equational Proofs

*(the Residual Perspective)*

*Ren e Vestergaard 1;2 Institut de Math ematiques de Luminy*

*Centre National de la Recherche Scienti que Marseille, France*

*James Brotherston 3*[*;*](#_bookmark0)*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 rst-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 -calculus: the resid- ual theory of 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 nite development property for the*

*-calculus. The proof has the same equational implications (e.g., con uence) 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 di erence 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.f*](mailto:vester@iml.univ-mrs.fr)*r, WWW:* [*http://iml.univ-mrs.fr/~veste*](http://iml.univ-mrs.fr/~vester)*r*

*3 Work undertaken while supported by LFCS, Edinburgh.*

*4 Email:* [*jjb@dai.ed.ac.uk*](mailto:jjb@dai.ed.ac.uk)*, WWW:* [*http://www.dai.ed.ac.uk/~jjb*](http://www.dai.ed.ac.uk/~jjb)

*c 2001 Published by Elsevier Science B. V. Open access under* [*CC BY-NC-ND license.*](http://creativecommons.org/licenses/by-nc-nd/3.0/)

*with binding (aka abstraction). The point of disagreement is one of syntactic* representation rather than any speci cs of proof methodologies. As for the pen-and-paper practices, rst-order abstract syntax with one-sorted variable names is a rather simple formalism that is well-suited to focus a reader's atten- tion 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 o ending 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.](#_bookmark49)

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

*( x:( y: x:xy)x)y*

*( y: x:xy)y*

*( x: z:zx)y*

*x:xy*

*z:zy*

*One branch requires the renaming of an x into, say, a z and, although the* resulting terms are -equivalent, they are not equal as rst-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 -[equivalent] are identi ed."*

*\2.1.13. If M1;::: ; Mn occur in a certain mathematical context, [their]* bound variables are chosen to be di erent from the free variables."

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

*In a formal setting, post xed name-uni cation 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 -con uence 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 diÆculties with* names and equational proofs over rst-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 rst-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 con icts between residuals of di erent 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' de nitions 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 rst author's [28] and the authors' [27].*

*6 On a personal note, we also nd that the methodology is helpful in analysing and decom- posing 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 nite development property (SFDP), cf. Section 6, for a calculus of linking with rst-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 -equivalence plus AC-equivalence over an inductively-de ned 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 - con uence proof. The mechanisation speci cs 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 rst author to prove*

*-con uence, -over- postponement, and -standardisation [26]. The rst* of these follows the pattern we account for here in being concerned with (hor- izontal) commutativity whereas the latter two are radically di erent in being concerned with distributivity, or vertical commutativity, of relations. 7

*1.6* [*This*](#_bookmark4) *Paper*

*Section 2 presents the basics of rst-order representations of higher-order lan-* guages. In Section 3, we establish that the residual theory of in the - calculus is renaming-free up-to an initiality condition. This means that a host of key lemmas for equational properties (of the -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 con uence in a non-standard way (i.e., without niteness 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 -calculus proper in a rst-order manner,* we enforce a distinction between raw and real calculi. The former are induc- tively de ned 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 distribu- tivity, 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 de nition pertaining to Abstract Rewrite Systems (ARS) | an ARS is a collection of binary relations on the same domain.

*De nition 2.1 Consider an ARS of the form: 9 9 Ks; 9 9 Kc A A (the raw* calculus). Its structural collapse is induced from 9 9 Kc by factoring out the equivalence relation generated by 9 9 Ks; the new, real, relation is written !c:

*As = A= ==s*

*b cs : A ! As*

*M 7! fN j M ==s Ng*

*bMcs !c bNcs , M ==s; 9 9 Kc; ==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

*rst is a minimal requirement: the raw and real equational theories coincide.*

*Proposition 2.2 Consider an ARS 9 9 Ks; 9 9 Kc A A and its structural col-* lapse.

*A= ==s = As===*

*[c c*

*The second gives a suÆcient criterion for the raw/real equivalence of not* only equational theories but of the ultimate equational property: [con uence.](#_bookmark9)

*Lemma 2.3 Consider an ARS 9 9 Ks; 9 9 Kc A A and its structural collapse. 8*

*0 bMcs !c bNcs , M 9 9 KKs[c N 1*



*)*

*@ ^ Con ( !c) , Con (9 9 Ks[c) A*

*The con uence 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 con uence 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 -reduction in the (raw) -calculus is renaming-free up-to an initiality condition. The implication is that all equa- tional reasoning about the relevant fragment of the residual theory of 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, re exive relations. Con is con uence.*

*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.

*y[x := e] =*

*8e if x = y*

*(e1 ? e2)[x := e] = e1[x := e] ? e2[x := e]*

*:y otherwise*

*( y:e0)[x := e] =*

*y:e [x := e] if x 6= y ^ y 62 FV(e)*

*0*

*: y:e*

*0*

*otherwise*

*FV(y) = fyg*

*FV(e1 ? e2) = FV(e1) [ FV(e2)*

*FV( y:e) = FV(e) n fyg*

*Captx(y) = ;*

*Captx(e1 ? e2) = Captx(e1) [ Captx(e2)*

*Capt ( y:e) =*

*8fyg[ Captx(e) if x 6= y ^ x 2 FV(e)*

*x*

*:;*

*otherwise*

*Fig. 1. Total but partially correct substitution, [ := ], free variables, FV( ),*

*res*

*operator ? is used to range over marked and ordinary application: @ and*

*and variables capturing free occurrences of x, Captx( ), for var. NB! The in x*

*.*

*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.

*De nition 3.1 [The Raw, Marked -Calculus] Terms are given by*

*var ::= x j var var*

*j x:**var*

*j ( x: var)@ var*

*res*

*res*

*res*

*res*

*res*

*res*

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

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

*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) -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 -rule for technical reasons (without it, Lemma 4.8 requires an initiality condition).

*FV(e2) \ Captx(e1) = ;*

*( res )*

*( x:e1)@ e2 9 9 K res e1[x := e2]*

*e 9 9 K*

*1*

*res e*

*0*

*1*

*e 9 9 K*

*2*

*res e*

*0*

*e ? e 9 9 K*

*0*

*(A1res )*

*2*

*1 2*

*res*

*e ? e*

*1*

*2*

*e ? e 9 9 K*

*1 2*

*res*

*e ? e*

*1*

*0*

*2*

*(A2res )*

*e 9 9 K res*

*x:e 9 9 K*

*e0*

*(Lres )*

*x:e0*

*res*

*e 9 9 Kp res e e 9 9 Kp res e FV(e ) \ Capt (e ) = ; x 2 FV(e )*

*0*

*0*

*0*

*0*

*0*

*1*

*1 2*

*2*

*2*

*x* *1*

*1*

*( resKp )*

*( x:e )@ e 9 9 Kp res e [x := e ]*

*0*

*0*

*1 2*

*1*

*2*

*e 9 9 Kp res e x 62 FV(e )*

*0*

*0*

*1*

*1*

*1*

*( )*

*res*

*Kp*

*( x:e )@ e2 9 9 Kp res e*

*1*

*0*

*1*

*lazy*

*(VarKp )*

*res*

*e 9 9 Kp res e*

*0*

*res*

*x 9 9 Kp res* *x*

*(LKp )*

*0*

*x:e 9 9 Kp res x:e*

*e 9 9 Kp res e*

*1*

*0*

*1*

*e 9 9 Kp res e*

*2*

*0*

*2 res*

*e e 9 9 Kp*

*0 0*

*(AKp )*

*1 2*

*res*

*e e*

*1 2*

*Fig. 2. Raw residual relations for the marked -calculus: one-step reduction and*

*residual-completion, respectively.*

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

*De nition 3.2 [Barendregt Conventional Form] A term is said to be a BCF* if each binder name is unique and di erent from the free variables in the term.

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

## *(BCF)*

*res*

 *Æ*

*Proof. By structural induction in the initial BCF. The only non-trivial case* is for ( x:e1)@ e2. It follows by monotonicity of bound and free (but not cap- turing) variables under 9 9 Kp res . The BCF-condition ensures that no changes to e1 can block the redex as capturing variables are included in bound variables.2

*Lemma 3.4 (Raw Completion of Residual )*

*res*



*Proof. By transitive, re exive induction, it suÆces to prove:*

*res*

*The proof is by rule induction in 9 9 Kp res . The interesting cases are ( ) and* ( res ). The proof follows from Substitutivity and Substitution Lemmas. 2

*res*

*lazy*

*Substitutivity and Substitution Lemmas are non-trivial to prove formally.* For our present purposes we will merely present one of each to give an indica- tion of the style. For the following lemmas, we note that Captx(e1)\FV(e2) =

*; is the weakest predicate ensuring the correctness of substituting e2 into e1*

*for x.*

*Lemma 3.5 (Marked Substitution)*

*y 62 FV(e2) ^ (Captx(e3) \ FV(e2) = ;) ^ (Capty (e1) \ FV(e3) = ;)*

*^ x 6= y ^ (Captx(e1) \ FV(e2) = ;) ^ (Captx(e1[y := e3]) \ FV(e2) = ;)*

*+*

*e1[y := e3][x := e2] = e1[x := e2][y := e3[x := e2]]* Lemma 3.6 ( Residual-Completion Substitutivity)

*0*

*0*

*e1 9 9 Kp*

*res e1*

*^ e2*

*9 9 Kp*

*res e2*

*^ (Capt (e ) \ FV(e )= ;) ^ (Capt*

*(e0 ) \ FV(e0 )= ;)*

*x 1* *2*

*+*

*x 1* *2*

*0 0*

*e1[x := e2] 9 9 Kp res e1[x := e2]*

*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 -Residual Theory is not Blocked)*

*res res*

## *(BCF)* *Æ*



*Proof. By Lemma 3.3, any BCF residual-completes. By Lemma 3.4, such a* completion can absorb any initial 9 9 KK res . *2*

*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 *rule)*

*can be contracted without resulting in a variable clash. We believe the lazy*

*-rule could be avoided for the purposes of the above result. The conclu-* sion would then be that all marked redexes can be contracted. However, this would greatly complicate the proof of Lemma 3.4 as the transitive, re exive induction no longer would be straightforward. The reason is that the consid- ered property would need to be formulated with, say, BCF-initiality which is not preserved along the inducted relation. As it stands, the only enforced re- striction is that 9 9 Kp res must be well-de ned 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 *calculus (see, e.g., [1, Theorem*

*11.2.25]).*

*De nition 4.1 Let ! be a residual relation, i.e., a relation which only con-* tracts marked redexes (such as 9 9 K res in Section 3).

*! is said to be the corresponding development relation*

*e ! e0 is said to be a development (of e)*

*a development, e ! e0 , is said to be complete if unMarked(e0 ). 10*

*De nition 4.2 [The SFDP] A relation, !, is said to enjoy the strong nite* development property if

*(i) SN(!)*

*| developments are nite*

*10 With the obvious de nition of unMarked( ).*

*(ii) e !! e0 ) (9e00 :e0 ! e00 ^ unMarked(e00 )) 11*

*y 62 Capt (e) [ FV(e)*

*x*

*e 9 9 Ki res e*

*y*

*0*

*y*

*x:e 9 9 Ki res y:e[x := y]*

*( res)*

*y*

*x:e 9 9 K*

*0*

*i res x:e*

*(Lres )*

*e1 9 9 Ki res e*

*y*

*0*

*1*

*e2 9 9 Ki res e*

*y*

*0*

*e ? e 9 9 K*

*y*

*0*

*(A1res )*

*2*

*y*

*0*

*(A2res )*

*1 2*

*i res e ? e*

*1*

*2*

*e ? e 9 9 K*

*1 2*

*i res e ? e*

*1*

*2*

*Fig. 3. The raw indexed res-relation.*

*| developments can be completed*

*(iii) (e !! e1 ^ e ! e2 ^ unMarked(ei)) ) e1 = e2*

*| completions are unique*

*Intuitively, developments are nite 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 niteness* of developments. In contrast to conventional wisdom, 12 we show that even without niteness of developments, the SFDP (that is, the S(F)DP) can be used to conclude con uence. The proof method is a hybrid of the standard SFDP method and the Tait/Martin-Lof/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' -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 -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-Lof con uence 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 commutativ- ity we use is between computational relations proper (e.g., res) and relations

*11 Observe that e0 ! e00 only can contract residuals of e as the terms are marked.*

*12 See, e.g., [1, p.283]: \[The niteness of developments] has important consequences, among them being [con uence] ..."*

*which axiomatise structural equivalence. In the raw, marked -calculus, the* relevant notion of structure is the res-relation, cf. Figure 3 (which like most other formal statements in this section, are adapted from [27]). The gure presents an indexed relation which is intended to smoothly facilitate the fol- lowing de nition.

*De nition 4.3 Following Figure 3,*

*res 0 y* *0*

*let the -relation be: e 9 9 K res e*

*res*

*, 9y:e 9 9 Ki res e*

*y*

*0 0 13*

*let fresh-naming 0 be: e 9 9 K res e*

*0*

*, 9y:e 9 9 Ki res e ^ y 62 Var(e)*

*We stress that the res-equivalence relation induced from the above 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 (9 9 K res -Symmetry)*

**

*Proof. By rule induction in the underlying indexed relation 9 9 Ki res . The only* non-trivial case follows by basic `renaming sanity' properties of substitution.2

*The reason for presenting two -relations is that a fully general commuta-* tivity result for res and 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 res-step will block a res-step.

*0*

*Lemma 4.5*



*Æ*



*Æ*

*Proof. By transitive, re exive induction in 9 9 KK res . The re exive and tran-* sitive cases are trivial. In the base case we proceed by rule induction in the

*0*

*underlying indexed relation: 9 9 Ki res . The proof is rather involved and uses the*

*0*

*relevant Substitutivity lemmas on 9 9 Ki res as well as a Substitution Lemma.*

*We also take advantage of the fact that it suÆces to use the same name for*

*each step of the 9 9 KKi res -closure.* *2*

*It is not diÆcult to see that 9 9 K res 9 9 K res . It can also be shown that*

*0*

*9 9 KK res 9 9 KK res . However, by the following lemma we have == res = == res .*

*0*

*Lemma 4.6*



*0*

*res*

## *Æ*

*13 With 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 suÆces to rename the two -equivalent terms with suÆ- ciently many fresh names in the same order in order to arrive at the same term. Formally, the proof proceeds by transitive, re exive induction (by Lemma 4.4, it suÆces to induct over 9 9 KK res ) and then by rule induction in 9 9 K res . The formal proof is complicated greatly by the need for explicit quanti cation over lists of variable names used for the reduction sequences. Substitutivity and Substitution Lemmas are also required. The proofs do not provide any in- sights and are omitted here; for a fuller exposition the reader is invited to consult [27]. *2*

*The reason for the above asymmetry between res and res is that not*

*0*

*all terms can be the target (although they naturally can be the source) of a*

*9 9 K res -step, whereas some terms are not too diÆcult to target:* Lemma *4.7*

*0*

*resÆ (BCF)*



*0*

*Proof. Intuitively, the property holds since renaming every -binder occur-*

*ring in the term with a fresh name will yield a BCF term. To prove the result* formally, we rst prove the following property by structural induction on e1:

*! !*

*8 zi; e1: jj zi jj= #(e1) ^ fzig all di erent ^ (fzig\ (FV(e1) [ BV(e1)) = ;)*

*+*

*!*

*zi*

*9e2:e1 9 9 KKi 0 e2 ^ BCF(e2) ^ fzig = BV(e2)*

*where #(e1) is the number of -abstractions in e1. The inclusion of the* variable name information is necessary in order to make the proof fully con- structive.

*2*

*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 -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 )*

*res*



*Proof. By transitive, re exive induction, it suÆces to prove the one-step case.*

*res*

*The following diagram outlines the proof of the de nitional re-statement of* the property at the raw level.

## *M1*

*l 2*

res

res

*M*



*l r*

0

*M N0 * *M*

*3 res* *2*

*0*



res

*l (BCF)N1*

*res*

res

0

*M*

*4*

*l r*

*N*

*M*

*2 3*

*res*



*r 2*

*res*

*N*

*r 4*

*M*

*For the Ms given, we* can construct the Ns in the divergence resolution on the left in order (0, 1, ... ). N0 follows from Lemma 4.6. N1 from Lemma 4.7. The N2's from Lemma *4.5.*

*The lower leg follows by*

*transitivity of == res and* Lemma 3.4.

*2*

*Lemma 4.9 (Real Residual-Completion) Æ*

*res*

*Proof. By De nition 2.1, we must prove: Æ * *Æ Æ*

*The result follows from Lemmas 4.7 and 3.3.*

*res res res* *2*

*Proposition 4.10 (Further Properties of Real Residual-Completion)*

*(i) !p res is functional*

*(ii) e !p res e0 ) unMarked(e0 ) | residual-completion leaves no marks*

*(iii) !p res !!res | residual-completion is a development*

*Proof. It essentially suÆces 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).*

*2*

*Theorem 4.11 !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 de nition. The resulting* term is unique by functionality of !p res (Proposition 4.10) and Lemma 4.8.2

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

# *5 Con uence from the S(F)DP*

*The original justi cation for considering residuals is the fact that develop-* ments (as opposed to unrectricted -reduction) can be proved nite [4,22]. By using Newman's Lemma, 14 the development relation can thus be proved con uent merely by proving it weakly con uent. In turn, this can be used to prove the -calculus proper con uent, as we will detail in Section 5.2.

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

*Lemma 5.1 (9 !b : !a !b !!a ^ (!b)) ) Con (!a)*

*Proof. A formalisation is provided in [21] and is re-used in [27].* *2*

*Lemma 5.2 (Guarded Diamond Diagonalisation) For any predicates, P* and Q, and any relations, !a and !b, we have

*(P )*

*Æ*

*b*

*(Q)*

*^*

*b*

## *(P ^ Q)*

## *) Æ*

*Proof. Straightforward. A formalisation is provided in [27] following [25]. 2*

*5.1 Residual Con uence*

*We believe the proof of the following result is new. We let it speak for itself.* Lemma 5.3 Con ( !res ) ^ Con (9 9 K res [ res )

*Proof. As for the rst 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 con uence and strong normalisation implies con uence.*

*15 See, e.g., [1, p.283]: \[The niteness of developments] has important consequences, among them being [con uence] ..."*

*by Lemma 4.9. The second result follows from the rst and Lemma 4.4 by* Lemma 2.3. *2*

*5.2 Con uence Proper*

*This section uses standard techniques and is provided virtually uncommented.*

*De nition 5.4 Let j j: var ! var be the function that removes marks* from a marked raw term (with the obvious de nition of var). Let

*res*

*j e1 j !dev j e2 j , e1 !res e2* be the non-marked development relation.

*def*

*Proposition 5.5 ! !dev !*

*Proof. Both inclusions follow by simole rule inductions at the raw level. 2* Theorem 5.6 Con ( !) ^ Con (9 9 K [ )

*Proof. As for the rst result, the !dev-relation is a witness of the quanti -* cation in Lemma 5.1 according to Lemma 5.3 and Proposition 5.5. The second result follows from the rst and an adaptation of Lemma 4.4 by Lemma 2.3.2

# *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 pre- sented. We stress that [28] introduces the m-calculus, a calculus of linking with rst-class primitive modules. The m-calculus is vastly more expressive than the -calculus in terms of the computational paradigms in can simu- late directly [28]. Among other things, it contains mutually recursive binding amongst sets of elements. For uniformity, we frame both results in terms of the *calculus.*

*6.1 The Strong Finite Development Property [28]*

*Apart from niteness of developments, it suÆces to prove a real residual rela-* tion weakly con uent to prove the SFDP and con uence. The latter step uses Newman's Lemma. The former is conducted at the raw level.

## *M0*

*e 9 9qK e*

*0*

*e 9 9qK e*

*0*

*1 1*

*e 9 9qK e*

*0*

*2 2*

*x 9 9qK* *x*

*x:e 9 9qK x:e*

*0*

*0 0*

*e e 9 9qK e e*

*1 2*

*1 2*

*e 9 9qK e*

*0*

*1*

*e 9 9qK e*

*2*

*0*

*2*

*FV(e ) \ Capt (e ) = ;*

*0*

*0*

*1*

*2*

*x* *1*

*( x:e )e 9 9qK e [x := e ]*

*0*

*0*

*1 2 1*

*2*

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



*Ml N0 Mr*

*The M-divergence on the left* is the re-statement at the raw level of a one-step divergence in

*1*  *0 0*

## *(BCF)*

0

*M*

*1 the real calculus according to* De nition 2.1. We have sup-

*l N1 Mr*



*2*

*2*



*l r*

*2 2*

*N*

*N*

 

*l N3 Mr*

*M*

*3*

*3*

*pressed the \res" superscripts*

*on the residual-relation names.* The N's are constructed in or- der (0, 1, . . . ) to resolve the di- vergence. Adaptations of Lem- mas 4.4, 4.5, 4.6, and 4.7 are easily recognised in the proof.

*6.2 Con uence via Parallel Reduction [27]*

*The Tait/Martin-Lof proof method uses a parallel relation that can contract* any number of pre-existing redexes in one step, 9 9qK below. The point is that such a relation can be shown to be a witness for the quanti cation in Lemma

*5.1. Takahashi [25] introduced the idea of de ning the parallel relation di-* rectly 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.

## *M*



*l r*



*M N0 M*

*1 0 0 1*



*l N1 (BCF)*

0

*M*

*2*

*The proof resolution tech-* nique on the left is assumed to be familiar to the reader at this point. Instead, we note that the proof estab- lishes (9 9 KK ; 9 9qK ) | the

*r symbol \;" is for relation* 2

*M*

*composition. As opposed*



*to the other two proofs*

*l r we have presented, we use*



*N*

*N*

*2 2 this property to prove (raw)*

*Con (9 9 K [ ) rst (Lemma* 5.1) and from there con-

*N3 clude (real) Con ( !) via*

*Lemma 2.3.*

# *7 Conclusion*

*We hope to have convinced our reader that the pen-and-paper proof prac-* tices of the wider programming language theory community are formalisable. In trying to do so, we have proved an approximation of the SFDP of the

*-calculus. It was observed that the presented results imply that the resid-*

*ual theory of -reduction is renaming-free up-to BCF-initiality. The results* also imply con uence. 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 con uence for a higher-order language represented with rst-order abstract syntax over one-sorted variable names. On-going work aims at ex- tending 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 r esultats classiques en calcul, Comptes Rendus de l'Acad emie des Sciences 320 (1995), pp. 1401{1406, s erie 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. Schurmann, 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 -calculus: A formal development, Journal of Functional Programming 4 (1994), pp. 371{394.*

*[16] Joachimski, F. and R. Matthes, Standardization and con uence 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 Godel'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 -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 rst-order con uence proof for the -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 rst-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 or Æ. Informally, this denotes quanti cation 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 Æ must be half-coloured. A diagram must be type-correct on domains. A property is read o of a diagram thus:*

*(i) write universal quanti cations for all s (over the relevant domains)*

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

*(iii) conclude the guarded existence of all Æs and their relations*

*The following diagram and property correspond to each other (for ! A A).*

*(P ) *

*Æ*

*(Q)*

*8e1; e2; e3 2 A: e1 ! e2 ^ e1 ! e3 ^ P (e1)*

*+*

*9e4 2 A: e2 ! e4 ^ e3 ! e4 ^ Q(e4)*

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