JAIST Repository
https://dspace.jaist.ac.jp/

Title	Structural preservation and reflection of diagrams
Author(s)	Norrish, Michael; Vestergaard, Rene
Citation	Research report (School of Information Science, Japan Advanced Institute of Science and Technology), IS-RR-2006-011: 1-12
Issue Date	2006-07-07
Type	Technical Report
Text version	publisher
URL	http://hdl.handle.net/10119/8410
Rights	
Description	リサーチレポート（北陸先端科学技術大学院大学情報科学研究科）

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

Structural Preservation and Reflection of Diagrams

Michael Norrish
Canberra Research Lab, NICTA
Canberra, Australia
Michael.Norrish@nicta.com.au

René Vestergaard[1]
School of Information Science, JAIST
Ishikawa, Japan
vester@jaist.ac.jp

IS-RR-2006-011

July 7, 2006

[1]This author gratefully acknowledges NICTA's financial support and hospitality while a visiting researcher.

Structural Preservation and Reflection of Diagrams

Michael Norrish
Canberra Research Lab, NICTA
Canberra, Australia
Michael.Norrish@nicta.com.au

René Vestergaard*
School of Information Science, JAIST
Ishikawa, Japan
vester@jaist.ac.jp

Abstract

We present a formal notion of diagram *that captures standard rewriting properties such as confluence, commuting relations, semi-standardisation, the triangle property, etc. We prove a number of results that imply* diagram equivalence *between abstractly related rewrite relations, i.e., that show that one enjoys a diagram-expressed property iff the other one does. In particular, we focus on the case where one rewrite system is over structurally collapsed terms of the other. We apply the results to various concrete systems, including the λ-calculus, with its notion of α-equivalence, and cut-elimination, with its notion of permutative conversions. The core theory and substantial applications have been formally verified in the HOL4 proof assistant.*

1 Introduction

This work makes four important contributions

- *Foundationally*, we want to answer the question: "how do different formalisations of the same subject-domain (e.g., the λ-calculus) inter-relate, and in particular do they entail the same behavioural theorems?" This question is highly pertinent to recent interest in languages with binders (see, for example, the POPLMARK Challenge [2]) because the proposed techniques for modelling binders (de Bruijn terms, quotients, higher order syntax, nominal approaches, and others) are rather differently construed and executed.
- Our results have *practical* relevance because they allow for a broad family of properties proved of one approach to be shown true of another in "one fell swoop".
- Our results also have *technical* relevance because, as we show in Section 6.2, our results allow us to prove properties that on the surface might look intractable.
- Finally, we contribute by formalising *diagrams* as a simple language for capturing the behavioural properties desired of a type. Our diagrams are given a formal semantics and allow us to focus on extensional behaviour without becoming entangled in intensional structure or composition. Moreover, we shall see that the derived notion of diagram equivalence is not so constrained as to force types to be isomorphic.

Choice of Language Our results are obtained through the study of epimorphisms between rewrite systems and say that (the enjoyment or not of) any possible diagram is preserved and reflected across a large class of epimorphisms. The specific language of properties captured by the diagrams is therefore of crucial importance. We can testify through personal experience that our diagram language is rather expressive but we leave for future work an objective characterisation of the issue; see, however, Section 3. Among the questions that can be addressed are: can the language of properties formalisable as a diagram be independently characterised (similar to the Goldblatt-Thomason Theorem in modal logic [8])? What is the largest class of functions that preserve and reflect diagrams? What is the largest set of properties that are preserved and reflected by our class of functions? As it stands, these issues are indirectly addressed by the facts that i) the use of diagrams is a wide-spread practice and ii) our formal development is surprisingly straightforward and fully algebraic.

Relations We consider Abstract Rewrite Systems (ARSs), $\rightarrow_x \subseteq X \times X$, over carrier set, X, with primitive equality, $=_X$. The reflexive, transitive (or pre-order) closure of an ARS, $\rightarrow_x$, is denoted $\twoheadrightarrow_x$ or x^*, depending on context. Symmetric closure is denoted $\hat{x}$, while reflexive, transitive, symmetric (or equivalence) closure is denoted $=_x$ or $\overline{x}$. Juxtaposition (e.g., xy) indicates relation-union. Relational composition is written with a semi-colon $x; y$.

*This author gratefully acknowledges NICTA's financial support and hospitality while a visiting researcher.

2 Diagrams

Diagrams with solid (universal) and dashed, dotted or grey (existential) lines abound in the rewriting literature. Barendregt [4, "Hints for the Reader"] calls them *"category theoretic" pictures*[1]. Baader and Nipkow use them and their "precise meaning" throughout their book [3]. Following Vestergaard [26], we say a commutative diagram of this nature is a set of coloured vertices and a set of coloured directed edges between pairs of vertices. Informally, the colour of a vertex (solid *vs* open) denotes quantification modes over terms, universal and existential, respectively. Edges are written as the relational symbol they pertain to and are either solid or dotted. Informally, the colour indicates assumed and concluded relations, respectively. An edge connected to an open circle must be dotted. A property is read from a diagram thus:

1. write universal quantifications for all solid circles
2. assume the solid relations
3. conclude existence of open circles and dotted relations

The following diagram and formula are thus equivalent:

$$\forall x\ y\ z.\ R^*(x,y) \wedge R^*(x,z) \Rightarrow \exists u.\ R^*(y,u) \wedge R^*(z,u)$$

The use of open circles as well as dotted lines allows us to distinguish *(onto)*:

(that is, $\forall y.\ \exists x.\ R(x,y)$) from *(complete)*:

(that is, $\forall x\ y.\ R(x,y)$).

A diagram is thus a graphical representation of a Π_1 condition on a relation (or a family of relations, as in commutativity and sequentialisation statements).

Definition 1 (Diagrams) *A diagram is a quadruple $\langle B, W, F, E\rangle$, with B the set of closed ("black") circles, and W the set of open ("white") circles. $F \subseteq \mathbb{N} \times B \times B$ represents the solid ("forall") links between solid circles, where the natural-number indexing serves to identify different relations (if necessary). Finally, $E \subseteq \mathbb{N} \times (B + W) \times (B + W)$ represents the dotted ("existential") links (again, possibly of different sorts), which may be between either solid or open circles.*[2]

[1] The difference is that dashed lines in category theoretic pictures are usually read as "unique existence", and that diagrams are concerned with equality of composed arrows.

[2] The use of $\mathbb{N}$ to index relations is an arbitrary choice: for added generality, diagrams could be parameterised by this index set. Further, though we will not exploit it, we note that this formalism allows diagrams of infinite size.

We can now capture what it is for a diagram to be true of a family of relations.

Definition 2 (Diagram Evaluation) *A diagram d is true of relations R_i over $X \times X$ (written $R_i \models d$), if for every function $f : B \to X$ that is homomorphic on all the F_i and R_i simultaneously (i.e., $\forall n\ b_1\ b_2.\ (n, b_1, b_2) \in F \Rightarrow (f(b_1), f(b_2)) \in R_n$), there is a $g : W \to X$, such that*

$$\begin{array}{l} \forall n\ b_1\ b_2.\ \ (n, b_1, b_2) \in E \Rightarrow (f(b_1), f(b_2)) \in R_n \\ \forall n\ b_1\ w_2.\ \ (n, b_1, w_2) \in E \Rightarrow (f(b_1), g(w_2)) \in R_n \\ \forall n\ w_1\ b_2.\ \ (n, w_1, b_2) \in E \Rightarrow (g(w_1), f(b_2)) \in R_n \\ \forall n\ w_1\ w_2.\ \ (n, w_1, w_2) \in E \Rightarrow (g(w_1), g(w_2)) \in R_n \end{array}$$

(Values b_i and w_i are implicitly injected into the appropriate half of the disjoint union when membership of E is asserted.)

Example: Diamond The graphical diamond property diagram

is formally captured by $Diamond = \langle\{0,1,2\}, \{0\}, F, E\rangle$, where

$$\begin{array}{rcl} F & = & \{(1,0,1),\ (1,0,2)\} \\ E & = & \{(1, \mathsf{inl}(1), \mathsf{inr}(0)),\ (1, \mathsf{inl}(2), \mathsf{inr}(0))\} \end{array}$$

Lemma 3 *The diagram Diamond is true of a family of relations R_i in the sense of Definition 2 iff R_1 has the diamond property, i.e.,*

$$\forall x\ y\ z.\ R_1(x,y) \wedge R_1(x,z) \ \Rightarrow\ \exists u.\ R_1(y,u) \wedge R_1(z,u)$$

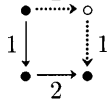
Example: Completeness The graph-completeness diagram

is captured by

$$GComp = \langle\{0,1\}, \emptyset, \emptyset, \{(1, \mathsf{inl}(0), \mathsf{inl}(1))\}\rangle$$

Lemma 4 *The diagram GComp is true of a family of relations R_i iff R_1 is complete, that is $\forall x, y.\ R_1(x,y)$.*

Example: Commuting Relations The commuting-relations diagram



is captured by $ComRel = \langle\{0,1,2\}, \{3\}, F, E\rangle$ where

$$\begin{array}{rcl} F & = & \{(1,0,1),\ (2,1,2)\} \\ E & = & \{(2, \mathsf{inl}(0), \mathsf{inr}(3)),\ (1, \mathsf{inr}(3), \mathsf{inl}(2))\} \end{array}$$

Lemma 5 *The diagram ComRel is true of a family of relations R_i iff R_1 and R_2 commute, i.e.,*

$$\forall x\ y\ z.\ R_1(x,y) \wedge R_2(y,z) \Rightarrow \exists u.\ R_2(x,u) \wedge R_1(u,z)$$

Note also that a diagram with only solid edges (including the empty diagram) is vacuously true.

Definition 6 (Diagram Equivalence) *Relation families $\rightarrow_x$ and $\rightarrow_y$ are* diagram equivalent *(written Δ-Eq($\rightarrow_x, \rightarrow_y$)) if for all diagrams D, $\rightarrow_x \models D$ if and only if $\rightarrow_y \models D$.*

(Often we will only be interested in one particular pair of relations, rather than a whole family.)

In what follows we will develop a theory establishing sufficient conditions to show diagram equivalence. We consider the situation of two ARSs with interrelated carrier sets, one *concrete*, C, and one *abstract*, A. The relationship between C and A is captured by some total and onto function, $\lfloor - \rfloor : C \rightarrow A$ (that parameterises this and later sections). This function will be constrained in the way in which it preserves and reflects reductions in the two carrier sets.

Definition 7 *Let $\rightarrow_{c_i} \subseteq C \times C$ and $\rightarrow_{a_i} \subseteq A \times A$ be i-indexed ARS families.*

$$\begin{array}{lcl}
(Pres^{c_i}_{a_i}) & \triangleq & \forall i\ c_1\ c_2.\ c_1 \rightarrow_{c_i} c_2 \Rightarrow \lfloor c_1 \rfloor \rightarrow_{a_i} \lfloor c_2 \rfloor \\
(aRefl^{c_i}_{a_i}) & \triangleq & \forall i\ c_1\ c_2.\ c_1 \rightarrow_{c_i} c_2 \Leftarrow \lfloor c_1 \rfloor \rightarrow_{a_i} \lfloor c_2 \rfloor \\
(sRefl^{c}_{a_i}) & \triangleq & \forall i\ a_1\ a_2. \\
 & & \quad a_1 \rightarrow_{a_i} a_2 \Rightarrow \\
 & & \quad \exists c_1\ c_2.\ \lfloor c_1 \rfloor = a_1 \wedge \lfloor c_2 \rfloor = a_2 \wedge \\
 & & \qquad c_1 \rightarrow_{c_i} c_2
\end{array}$$

'Pres' stands for preservation, *'sRefl' for* some-reflection, *'aRefl' for* any-reflection; *we write $(Pres^{c}_{a})$ for $\forall i\ .\ (Pres^{c_i}_{a_i})$, and, analogously, write $(sRefl^{c}_{a})$ and $(aRefl^{c}_{a})$. Some authors identify functions satisfying $(Pres^{c}_{a})$ and $(aRefl^{c}_{a})$ as "strong homomorphisms".*

Our key lemma generalises earlier results [27] but it is worth noting that the arrived-at conclusion is very strong, indeed (although, it remains to be seen exactly how strong).

Lemma 8 (Any-Reflected Diagram Equivalence)

$$(Pres^{c}_{a}) \wedge (aRefl^{c}_{a}) \Rightarrow \Delta\text{-}Eq(\rightarrow_c, \rightarrow_a)$$

Proof Consider an arbitrary diagram $D = \langle B, W, F, E \rangle$. Case 1: D is true of $\rightarrow_c$, and an arbitrary $f : B \rightarrow A$ is homomorphic over F. As $\lfloor - \rfloor$ is onto, there is a right-inverse $h : A \rightarrow C$. Combining, we have that $h \circ f$ is homomorphic onto C by (aRefl^c_a). Because D is true of $\rightarrow_c$, there is a homomorphic $g : W \rightarrow C$. By (Pres^c_a), $\lfloor g(-) \rfloor$ is a homomorphism of the desired form from W to A.

Case 2: D is true of $\rightarrow_a$, and an arbitrary $f : B \rightarrow C$ is homomorphic over F. By (Pres^c_a), $\lfloor f(-) \rfloor$ is homomorphic over F into A. As D is true of $\rightarrow_a$ there is a homomorphic $g : W \rightarrow A$. By composition with an inverse of $\lfloor - \rfloor$, and use of (aRefl^c_a), we derive a homomorphism of the desired form from W to C. □

3 Modalities and Diagrams

As briefly discussed in Section 1, our results appear to be related (but complementary) to classic results in modal logic. It is instructive to look at some of the details of this connection in order to understand some of the subtleties of our diagram language, e.g., in relation to equality and equivalence, as such differences will become important later on.

3.1 Modal Frame Axioms

Diagrams are capable of expressing many of the notions over relations that are also expressible in the frame axioms of propositional modal logic. In addition to the diamond and commuting relations diagrams, we have the example of symmetry: the diagram is

while the frame axiom is

$$p \Rightarrow \Box\Diamond p$$

While there is considerable interesting overlap, it is equally clear that neither language contains the other. Not all modal formulas are expressible in first order logic, for example, whereas all diagrams are first order. Conversely, consider the following diagram for "existence of a reflexive point":

This property is not expressible as a frame axiom because it is not preserved by generated sub-frames, as required. In fact, diagrams are not guaranteed to be preserved by the three standard modal constructions:

Bounded p-morphisms: If ϕ is true of relation $\rightarrow_c$ on C, and there is a homomorphic, onto, function $f : C \rightarrow A$, where $f(c) \rightarrow_a a$ in A implies there exists a c' such that $f(c') = a$ and $c \rightarrow_c c'$, then ϕ is true of $\rightarrow_a$.

Disjoint unions: If every frame in a family R_i satisfies some formula ϕ, then so too does $\uplus_i R_i$.

Generated sub-frames: If ϕ is true of a relation R, then it is also true of the sub-relation R', where the domain of R' is closed under the relation.

The counter-example to diagram-preservation for bounded p-morphisms corresponds to the formula

$$\forall x\, y.\ R(x,y) \wedge R(y,x) \Rightarrow \exists z.\ R(x,z) \wedge R(z,y)$$

where the range of the p-morphism is a two-element loop (which falsifies the diagram), and where the domain is a three element chain (which satisfies the diagram).

The counter-example for disjoint unions is the diagram for a complete graph. While two graphs may be complete, their disjoint union will not be. Finally, we have already seen that the existence of a reflexive point is not preserved by generated sub-frames.

3.2 Equality

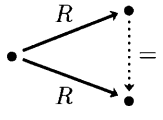
Diagrams are not able to capture all the first-order definable modal formulas, at least not without outside help. The simplest counter-example is determinism

$$\Diamond p \Rightarrow \Box p$$

or in its first order form

$$\forall x\, y\, z.\ R(x,y) \wedge R(x,z) \Rightarrow (y = z)$$

Without seeming to mention the notion at all, the modal formula refers to equality. It is not possible to similarly use equality implicitly inside diagrams. On the other hand, it is certainly possible to explicitly state that a diagram is being evaluated with respect to a family of relations, one of which is equality. Graphically, determinism could be done thus



To be explicit about the formal presentation, the above diagram is $\langle \{0,1,2\}, \emptyset, F, E\rangle$ where

$$\begin{aligned} F &= \{(0,0,1),(0,0,2)\} \\ E &= \{(1,\mathsf{inl}(1),\mathsf{inl}(2))\} \end{aligned}$$

The labelling of the arrows in the graph corresponds to asserting that the diagram is true of the relations $\langle R, =\rangle$ (thus, the F-links belong to relation 0, which is R, and the one E-link belongs to relation 1, which is equality).

Unfortunately, the strong homomorphisms that are the basis for our preservation and reflection results only preserve equality; they do not reflect it. Similarly, in modal logic, frame axioms are only preserved by bounded p-morphisms, not reflected by them.

In what follows, we successfully deal with equality by reflecting it back to equivalence. Complementing this approach, we shall return to "internalised" equality in Section 7.3.

4 Structural Collapse

Let us assume the concrete carrier set C contains undesired *structure*, axiomatised by $\rightarrow_s \subseteq C \times C$; A is intended to be a structure-free version of C.

Definition 9 *A is the* s-collapse *of C, $(Coll^C_A)_s$, for $\rightarrow_s$ if*

$$\begin{aligned} (kSound)_s &\triangleq \forall c_1, c_2.\ c_1 \rightarrow_s c_2 \Rightarrow \lfloor c_1 \rfloor =_A \lfloor c_2 \rfloor \\ (kCompl)_s &\triangleq \forall c_1, c_2.\ c_1 =_s c_2 \Leftarrow \lfloor c_1 \rfloor =_A \lfloor c_2 \rfloor \\ (Coll^C_A)_s &\triangleq (kSound)_s \wedge (kCompl)_s \end{aligned}$$

The first two properties state that $\rightarrow_s$ axiomatises the kernel of $\lfloor - \rfloor$: 'kSound' stands for kernel soundness *and 'kCompl' for* kernel completeness.

Definition 10 *We call $\rightarrow_s$* orientation-free *if any sequence of s-steps can be undone by further s-steps. (This is slightly weaker than simply requiring $\rightarrow_s$ to be symmetric.) We refer to the s-collapse of an orientation-free relation, s, as a* structural collapse.

$$\begin{aligned} (oFree)_s &\triangleq \forall c_1\, c_2.\ c_1 =_s c_2 \Rightarrow c_1 \rightarrow_s^* c_2 \\ (StrColl^C_A)_s &\triangleq (Coll^C_A)_s \wedge (oFree)_s \end{aligned}$$

With this, we note that we typically define $a_1 \rightarrow_a a_2$ to be $c_1 \rightarrow_c c_2$, for $a_i = \{c_i' \mid c_i' =_s c_i\}$. When we do this, we have, e.g., (Pres^c_a) and (sRefl^c_a) by construction, which motivates the following definition.

Definition 11 *$\rightarrow_a \subseteq A \times A$ is induced as the* (structural) collapse *of $\rightarrow_c \subseteq C \times C$, relative to $\rightarrow_s \subseteq C \times C$ if*

$$\begin{aligned} (Coll^{\rightarrow_c}_{\rightarrow_a})_s &\triangleq (Coll^C_A)_s \wedge (Pres^c_a) \wedge (sRefl^c_a) \\ (StrColl^{\rightarrow_c}_{\rightarrow_a})_s &\triangleq (StrColl^C_A)_s \wedge (Pres^c_a) \wedge (sRefl^c_a) \end{aligned}$$

Proposition 12

$$(StrColl^{\rightarrow_c}_{\rightarrow_a})_s \quad \Leftrightarrow \quad (Coll^{\rightarrow_c}_{\rightarrow_a})_s \wedge (oFree)_s$$

Proof By definition. □

We shall now attempt to lower the proof burden for uses of our key Lemma 8. First we address the any-requirement of the last property. In the presence of structure, we have a some-any equivalence:

Proposition 13 (Reflection is Structurally Some/Any)

$$(kSound)_s \wedge (kCompl)_s \Rightarrow ((sRefl^c_a) \Leftrightarrow (aRefl^{\overline{s};c;\overline{s}}_a))$$

Proof By two direct arguments. □

Due to its inherent any-nature, the preservation property is straightforwardly closed under the additional structure (use of $=_s; c; =_s$ instead of c):

Proposition 14 (Structural Preservation)

$$(kSound)_s \wedge (Pres^c_a) \Rightarrow (Pres^{\overline{s};c;\overline{s}}_a)$$

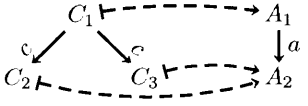
Proof By a direct argument. □

Thus, our first weakening result is that diagram-equivalence holds if we have preservation and some-reflection (rather than Lemma 8's any-reflection), but that the concrete reduction relation must be bracketed by s-equivalence steps:

Theorem 15 $(Coll^{\rightarrow_c}_{\rightarrow_a})_s \Rightarrow \Delta\text{-}Eq((=_s;\rightarrow_c;=_s),\rightarrow_a)$

Proof Lemma 8 and Propositions 13 and 14. □

Example: While Lemma 8 may make diagram equivalence look simple, it does, slightly counter-intuitively, establish the property between $\rightarrow_c$ and $\rightarrow_a$ below [27]. In passing, we note that this example demonstrates how the "determinism axiom" ($\Diamond p \Rightarrow \Box p$) from Section 3 is not reflected by strong homomorphisms.



Up-front, this may be taken to imply that the language of properties expressible as diagrams is weak, as it cannot separate the two relations. However, it is probably more reasonable to conclude that the involved issues are non-trivial and subtle. In particular, adding a reflexive $\rightarrow_a$-step on A_2 breaks (aRefl^x_y) because it necessitates $\rightarrow_c$ (or $\rightarrow_s$) steps between C_2 and C_3. The issues take shape when we pursue reflexive, transitive closures next. Indeed, considering relations that are reflexive and transitive appears to rule out anomalies like the one above, leaving us with a clean notion of diagram equivalence (but the issue remains open).

5 Pre-Order Reduction

When we consider the pre-order, or transitive, reflexive closures of our reduction relations (as is necessary for confluence, for example), structure again turns out be intertwined with the process of weakening "any" (aRefl) to "some" (sRefl) in Lemma 8. First, we note that preservation is straightforwardly closed under pre-ordering.

Proposition 16 (Pre-ordered Preservation)

$$(Pres^x_y) \Rightarrow (Pres^{x^*}_{y^*})$$

Proof By reflexive, transitive induction. □

But if we wish to include the structural relation s, the natural extension at the concrete level is to the relation $\rightarrow\!\!\!\!\rightarrow_{sc}$, i.e., $(s \cup c)^*$. This is not the reflexive and transitive closure of $(\overline{s};c;\overline{s})$, as the latter cannot do s by itself.

Lemma 17 (Structural Pre-ordered Preservation)

$$(kSound)_s \wedge (Pres^c_a) \Rightarrow (Pres^{(sc)^*}_{a^*})$$

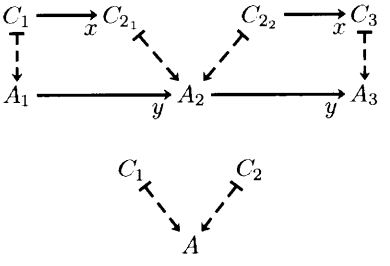
Proof By reflexive, transitive induction. □

The dual situation of pre-order closure for reflection is slightly different because the considered $\lfloor - \rfloor$ need not and typically will not be one-to-one.

Proposition 18

1. $(sRefl^x_y) \not\Rightarrow (sRefl^{x^*}_{y^*})$
2. $(aRefl^x_y) \not\Rightarrow (aRefl^{x^*}_{y^*})$

Proof The first property needs onto-ness for the reflexive case but fails in the transitive case. For the second property, the reflexive case is problematic while onto-ness makes the transitive case go through. Induced counter-examples:



□

With explicit s-equivalence, we have a similar result to Proposition 16 for (sRefl^c_a), provided we are considering a structural s. The relation at the concrete level is again $(s \cup c)^*$.

Lemma 19 (Some-Reflection is Structurally Pre-ordered)

$$(kCompl)_s \wedge (oFree)_s \wedge (sRefl^c_a) \Rightarrow (sRefl^{(sc)^*}_{a^*})$$

Proof By reflexive, transitive induction. □

Moreover, a some/any-equivalence holds.

Lemma 20 (Pre-ordered Structural Reflection is Some/Any)

$$(kCompl)_s \wedge (oFree)_s \Rightarrow ((sRefl^{(sc)^*}_{a^*}) \Leftrightarrow (aRefl^{(sc)^*}_{a^*}))$$

Proof By two direct arguments. □

The two preceding lemmas therefore give us the best we could hope for, namely that simple, computational-only some-reflection suffices for showing full-scale structural any-reflection.

Lemma 21

$$(kCompl)_s \wedge (oFree)_s \wedge (sRefl^c_a) \Rightarrow (aRefl^{(sc)^*}_{a^*})$$

Proof By Lemmas 19 and 20. □

We note that some-reflection allows for structure and computation to be addressed separately. In particular, structure need only be addressed once for each language.

To conclude:

Theorem 22 *$\twoheadrightarrow_{sc}$ and $\twoheadrightarrow_a$ are diagram equivalent if $\rightarrow_a$ is induced as the structural collapse of $\rightarrow_c$, relative to $\rightarrow_s$:*

$$(StrColl^{\rightarrow_c}_{\rightarrow_a})_s \quad \Rightarrow \quad \Delta\text{-}Eq(\twoheadrightarrow_{sc}, \twoheadrightarrow_a)$$

Proof By Lemma 8, it remains to be proved that we have $(\mathrm{Pres}^{(sc)^*}_{a^*})$ and $(\mathrm{aRefl}^{(sc)^*}_{a^*})$, which follow from Lemmas 17 and 21. □

5.1 Intermediate Relations

For technical reasons involving, e.g., parallel reduction relations, we note that we have the following result.

Theorem 23

$$\begin{array}{l}(StrColl^{\rightarrow_{c'}}_{\rightarrow_{a'}})_s \\ \wedge\, (\rightarrow_c \subseteq \rightarrow_{c'} \subseteq \twoheadrightarrow_{sc}) \;\wedge\; (\rightarrow_a \subseteq \rightarrow_{a'} \subseteq \twoheadrightarrow_a) \\ \Downarrow \\ \Delta\text{-}Eq((=_s; \rightarrow_{c'}; =_s), \rightarrow_{a'}) \;\wedge\; \Delta\text{-}Eq(\twoheadrightarrow_{sc}, \twoheadrightarrow_a)\end{array}$$

Proof Proposition 24, next, with Theorems 15 and 22. □

The outstanding result in the above proof concerns universality of reflexive, transitive closure for related relations.

Proposition 24

- $\rightarrow_a \subseteq \rightarrow_{a'} \subseteq \twoheadrightarrow_a \;\Rightarrow\; \twoheadrightarrow_{a'} = \twoheadrightarrow_a$
- $\rightarrow_c \subseteq \rightarrow_{c'} \subseteq \twoheadrightarrow_{sc} \;\Rightarrow\; \twoheadrightarrow_{sc'} = \twoheadrightarrow_{sc}$

Proof The first result is a special-case of the second result, with $\rightarrow_s$ the identity relation. The second result follows by two reflexive, transitive inductions, where only the base cases are interesting. □

6 Applications

Here we describe how the general theory presented so far can be used to demonstrate connections between different views of the same underlying behaviour. We also note that such connections do not need to be shown by the construction of a single homomorphism satisfying the constraints of Theorems 15 or 22. Instead, the fact that diagram-equivalence is an equivalence means that equivalences independently established with distinct homomorphisms can be stitched together to establish the diagram equivalence of superficially disparate systems. Indeed, we shall now discuss the diagram equivalences indicated in Figure 1: λJ^m and $\lambda\mathcal{G}$ are the (non-formalist) term calculi for cut-elimination

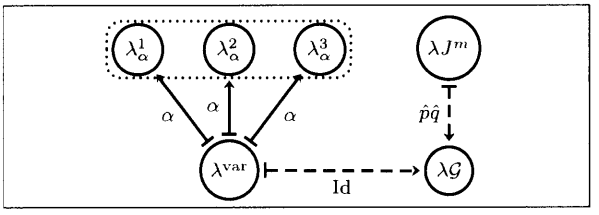


Fig. 1. Diagram-equivalent systems and their witnessing epimorphisms, indicated by collapsed structure; solid lines: formalised, dashed lines: discussed

studied in [21, 23, 22, 24]; λ^i_α are three different formalist presentations of the λ-calculus over α-collapsed terms (identified below in Section 6.1); and λ^{var} is the equally formalist version over first-order abstract syntax with one-sorted variable names and with explicit α in [26, 27].

6.1 The λ-calculus

In this section, we describe the connection between two different presentations of the λ-calculus: the "raw" first order syntax, and the same syntax quotiented by α-equivalence. The raw syntax is simply the free algebra generated by the recursion equation

$$\Lambda \;\cong\; V + \Lambda \times \Lambda + V \times \Lambda$$

with V some infinite set of variable names.

This type's behaviour (β-reduction) can be defined in at least two ways.

Partial Make substitution of M for variable v in term N behave correctly only when the abstractions in the term N do not need to be renamed to avoid capture of free variables in M. The β-reduction relation for this type can then be partial as well: if the preconditions for the β-reduction's substitution are not met, no reduction takes place.

The advantage of this approach is that the substitution function can be defined by primitive recursion. All desired reductions can be performed by first making a series of α-renaming steps. This approach is carefully presented by Vestergaard and Brotherston [26, 27].

Total Define substitution on Λ as a total function that may perform additional renaming steps as it passes through abstractions. Such a function can be defined by recourse to a definition using simultaneous substitutions, iterated substitutions, or well-founded recursion on the size of the argument. In our view, the advantages of totality are more than counter-balanced by the disadvantages of ugly definitions and the need to choose arbitrary fresh names as a term is traversed.

Either form of substitution on Λ allows the definition of α-equivalence. We can then take the quotient with respect to that equivalence, generating a fresh type that we will call Λ_α. Defining β-reduction on the quotiented type Λ_α can also be done in a number of ways.

Lifting substitution Homeier [11] defines a total substitution function on Λ and shows that this function respects α-equivalence, enabling it to be "lifted" to the level of Λ_α. Being able to do this is perhaps the best argument for using the "total" option above for substitution on Λ. With substitution defined on Λ_α, defining β-reduction is straightforward.

Lifting β-reduction One can define the β-reduction relation on Λ_α by reference to the relation on Λ, allowing for explicit α-conversion at the raw level [26, 27]. This avoids the need to define substitution at the level of Λ_α.

Defining substitution directly If one has a recursion principle for Λ_α justifying the definition of functions in a primitive recursive style, substitution can be defined directly. Recursion principles of the desired sort are discussed in papers by Ambler *et al.* [1], Norrish [16] and Pitts [18]. Then, with substitution defined at the level of Λ_α, β-reduction can also be defined directly.

With β-reduction defined on Λ_α it is possible to compare the two types and their associated behavioural relations. Below, we will write $\rightsquigarrow_\beta$ for the relation on Λ and $\rightarrow_\beta$ for the relation on Λ_α.

Clearly, the desired homomorphism between the two types must be the function taking a raw value in Λ to its equivalence class in Λ_α. We will write $\lfloor t \rfloor$ to denote the equivalence class of $t \in \Lambda$. By Theorem 15, there is diagram equivalence between β-reduction on Λ_α and $=_\alpha; \rightsquigarrow_\beta ; =_\alpha$ on Λ if it is possible to show $(\mathrm{Coll}^{\rightarrow_\beta}_{\rightarrow_\beta})_{\overline{\alpha}}$.

Because Λ_α is the quotient of Λ with respect to α-equivalence, we have a total, onto function with $(\mathrm{kSound})_{\overline{\alpha}}$ and $(\mathrm{kCompl})_{\overline{\alpha}}$ by definition. The remaining proof obligations are that the map from Λ to Λ_α must preserve and some-reflect reductions.

Lemma 25 (Preservation of β-reduction) *If $t \rightsquigarrow_\beta u$, then $\lfloor t \rfloor \rightarrow_\beta \lfloor u \rfloor$.*

Proof If β-reduction on Λ_α has been defined by lifting the operation from Λ, this result is immediate, by definition. Otherwise, the proof is by rule induction on the definition of $\rightsquigarrow_\beta$. The only interesting case is showing that substitutions in Λ can be matched by those in Λ_α. This proof is in turn straightforward because the preconditions from Λ require an appropriate choice of bound name in the abstraction. □

Lemma 26 (Some-reflection of β-reduction) *If $M \rightarrow_\beta N$, then there exist $t, u \in \Lambda$, with $\lfloor t \rfloor = M$ and $\lfloor u \rfloor = N$, such that $t \rightsquigarrow_\beta u$.*

Proof Again, if β-reduction on Λ_α has been defined by lifting the operation from Λ, then the result is immediate. Otherwise, the proof is by rule induction on the definition of $\rightarrow_\beta$. The congruence cases for applications require use of the fact that $\lfloor _ \rfloor$ is onto. The base case also uses this result, as well as the fact that it is always possible to find α-equivalent versions of terms that have their bound names chosen so as to avoid clashes when a substitution is performed. □

To show that confluence (a property of pre-order closures) at the raw level is present if and only if it is present at the quotiented level, we appeal to Theorem 22. This requires only that we now show $(\mathrm{oFree})_{\overline{\alpha}}$. As α-equivalence is indeed an equivalence, this is immediate. More, we have not just established equivalence for confluence, but that $\rightarrow^*_\beta$ and $\rightsquigarrow^*_{\overline{\alpha}\beta}$ share all possible diagrams.

We have formally verified the existence of structural collapses for three different formalist presentations of the λ-calculus over α-collapsed terms relative to λ^{var}.

λ^1_α: terms-as-α-equivalence-classes with "lifted β-reduction" [27, in Isabelle/HOL].

λ^2_α: Gordon and Melham's directly defined Λ_α [9], using a similarly directly defined β-relation [17, in HOL4].

λ^3_α: a quotient with substitution and β-reduction defined directly [17, in HOL4].

These systems are the various λ^i_α in Figure 1 and, by this paper and its HOL4-verified development, we therefore have diagram-equivalence of all these systems and λ^{var}.

In future work, there are at least two other λ-calculus systems that we would like to relate to λ^{var} in the same way: terms using de Bruijn indices, and a weak HOAS style presentation with abstractions represented using functions.

6.2 Structural Proof Theory

It is known that β-reduction in the λ-calculus is basically the same as Prawitz-style normalisation in a natural-deduction, or N-system, presentation of intuitionistic logic [12, 20]. Other presentations of intuitionistic logic are also possible, notably logical-deduction, or L-system, style using a cut rule (aka sequent calculus) [7]. There are compelling reasons why also β-reduction/Prawitz-style normalisation and Gentzen-style cut-elimination should be related [15]. Indeed, the connection was first explored in [30] and then presented in an algebraic form consistent with our notion of structural collapse in [19, "Normalization as a homomorphic image of cut-elimination"]. The relevant notion

of structure is axiomatised by what is referred to as *permutative conversions* [14] (covering also explicit-substitution [28] and pattern-matching issues [5] in their most general form). Although algebraically adequate, the early treatments of the result were not sufficiently fine-grained to allow for a non-structural, or computational, understanding of (all) permutative conversions. A main reason is that application in L-systems is best thought of as taking place "outside-in", as opposed to the "inside-out" of N-systems [10, 21] and of being of a generalised nature [29]. Following this realisation, permutative conversions have been reworked [6, 25] and a tight but staged correspondence has finally been arrived at [23, 22, 24]. We will now apply our framework to the considered systems and establish their diagram equivalence; we consider term languages, only, and refer to [23, 22, 24] for the proof/typing rules they capture.

6.2.1 Preliminaries

We first note that Gentzen [7] proved that his L-systems were not more expressive than his N-systems by exhibiting a (conclusion-respecting) mapping from the latter in to the former. In the case of intuitionistic implication, the image of the mapping is as follows.

Definition 27 ($\lambda\mathcal{G}$-Terms [21])

$$\begin{array}{rcl} \mathcal{G} & ::= & V \mid \lambda V.\mathcal{G} \mid \mathcal{G}(\mathcal{G}, L^{\mathcal{G}}, (z)z) \\ L^{\mathcal{G}} & ::= & [] \end{array}$$

The third clause in $\mathcal{G}$, e.g., $g_1(g_2, [], (z)z)$, captures Gentzen's target term for implication elimination, namely a cut on a left-introduction over an axiom. A term-model annotation for a general left-introduction of implication is let $x := f\ e_a$ in e_c [28]. The reading is that f is the introduced assumption of an implication; it takes the premise that typically sits on the left, e_a, and converts it into a witness that can be used to discharge the assumption, x, in the premise that typically sits on the right, e_c. The term e_c is the context, or continuation, which in the above rule simply is the identity $(z)z$. Another way of writing $g_1(g_2, [], (z)z)$ is therefore as (let $z{:=}f\ g_2$ in $z)[\![f{:=}g_1]\!]$, with "$[\![f{:=}_]\!]$" standing for a cut against the assumption denoted by f. Returning to the notation above, we note that the $L^{\mathcal{G}}$-category, which for now consists of the empty list, [], only, is due to Herbelin [10] and, in the general case, can be used to stack left-implication rules on top of each other. This is what amounts to "outside-in" application in L-systems because the function-position, i.e., the term that sits outside the parentheses, gets applied to g_2 inside the parentheses and then to any gs in $L^{\mathcal{G}}$, in the general case.

As it turns out, $\mathcal{G}$ is closed under cut-elimination and, following [23], we write $g_1[g_2]$ for $g_1(g_2, [], (z)z)$.

Definition 28 ($\lambda\mathcal{G}$-Reduction [21]) *Let* s *be meta-level substitution (of g_2 for x in g_1) and let* $\lambda\mathcal{G}$-reduction, $\rightarrow_{\beta\mathcal{G}}$, *be the contextual closure of the following contraction rule.*

$$(\lambda x.g_1)[g_2] \quad \rightarrow_{\beta\mathcal{G}} \quad \mathbf{s}(g_2, x, g_1)$$

$\lambda\mathcal{G}$ is, thus, a "notational variation" on the λ-calculus [21].

Lemma 29 $(StrColl^{\rightarrow_\beta}_{\rightarrow_{\beta\mathcal{G}}})_{\mathrm{Id}_\Lambda}$

Proof [21, Theorem 6, Chapter 5] shows that Gentzen's mapping is an isomorphism. □

Theorem 30 $\Delta\text{-}Eq(\rightarrow_\beta, \rightarrow_{\beta\mathcal{G}}) \ \wedge\ \Delta\text{-}Eq(\twoheadrightarrow_\beta, \twoheadrightarrow_{\beta\mathcal{G}})$

Proof Theorems 15 and 22 with Lemma 29. □

6.2.2 "A calculus of multiary sequent terms" [23, 24]

As suggested above, a fuller computational account of cut elimination is possible when removing the two artificial restrictions on $\lambda\mathcal{G}$-terms.

Definition 31 (λJ^m-Terms [23])

$$\begin{array}{rcl} J^m & ::= & V \mid \lambda V.J^m \mid J^m(J^m, L^{J^m}, (V)J^m) \\ L^{J^m} & ::= & J^m :: L^{J^m} \mid [] \end{array}$$

The exact differences between J^m and $\mathcal{G}$ are i) in the third clause, e.g., $j_1(j_2, l, (z)j_3)$, the context (in the left rule), j_3, need not be vacuous and ii) left implications can be stacked, j_2, l, to form a "multiary" application of j_1.

Definition 32 (λJ^m-Reduction [23]) *Let* s *and* @ *be meta-level substitution and append functions and let* λJ^m-reduction, $\rightarrow_{\beta_1\beta_2\pi\mu}$, *be the contextual closure of the following contraction rules.*

$$\begin{array}{l} (\lambda x.j_1)(j_2, [], (y)j_3) \\ \qquad \rightarrow_{\beta_1} \quad \mathbf{s}(\mathbf{s}(j_2, x, j_1), y, j_3) \\ (\lambda x.j_1)(j_2, j_0 :: l, (y)j_3) \\ \qquad \rightarrow_{\beta_2} \quad \mathbf{s}(j_2, x, j_1)(j_0, l, (y)j_3) \\ j_1(j_2, l, (y)j_3)(j'_2, l', (y')j'_3) \\ \qquad \rightarrow_{\pi} \quad j_1(j_2, l, (y)j_3(j'_2, l', (y')j'_3)) \\ j_1(j_2, l, (y)y(j'_2, l', (y')j'_3)) \\ \qquad \rightarrow_{\mu} \quad j_1(j_2, @(l, j'_2, l'), (y')j'_3) \end{array}$$

In the μ-rule, y must be fresh with respect to j'_2, l', j'_3 [25].

Definition 33 (λJ^m-Permutative Conversions [23]) *Let* $@^2$ *be a meta-level append-and-apply function and let the*

permutative conversion *relation,* $\rightarrow_{pq}$*, be the contextual closure of the following contraction rules.*

$$
\begin{array}{ll}
& j_1(j_2, l, (x)y) \\
\rightarrow_p & y \\
& j_1(j_2, l, (x)\lambda y.j_3) \\
\rightarrow_p & \lambda y.j_1(j_2, l, (x)j_3) \\
& j_1(j_2, l, (x)j_1'(j_2', l', (y)j_3')) \\
\rightarrow_p & j_1(j_2, l, (x)j_1') \\
& \quad (j_1(j_2, l, (x)j_2'), @^2(j_1, j_2, l, x, l'), (y)j_3') \\
& j_1(j_2, j_0 :: l, (x)j_3) \\
\rightarrow_q & j_1[j_2](j_0, l, (x)j_3)
\end{array}
$$

The main results in [23, 22], bar one, are as follows.

Theorem 34 ([23]) $\rightarrow_{pq}$ *is strongly normalising and confluent.*

Theorem 35 ([22, 24]) $\rightarrow_{\beta_0\mu\pi}$ *is strongly normalising and* $\rightarrow_X$ *is confluent for* $X \subseteq \{\beta_0, \mu, \pi\}$*, with* $\beta_0 = \beta_1\beta_2$*.*

6.2.3 Permutative Convertibility

The following function from J^m to $\mathcal{G}$ provides a crucial step in the above, as well as a further interesting property.

Definition 36 (Permutative-Conversion Collapse [23])

$$
\begin{array}{rcl}
\phi(x) & = & x \\
\phi(\lambda x.j) & = & \lambda x.\phi(j) \\
\phi(j_1(j_2, l, (y)j_3)) & = & \phi'(\phi(j_1), \phi(j_2), l, y, \phi(j_3))
\end{array}
$$

$$
\begin{array}{rcl}
\phi'(g_1, g_2, [], y, g_3) & = & \mathbf{s}(g_1[g_2], y, g_3) \\
\phi'(g_1, g_2, j :: l, y, g_3) & = & \phi'(g_1[g_2], \phi(j), l, y, g_3)
\end{array}
$$

Because of Theorem 34, we have a total function, $\downarrow_{pq}$, sending J^m-terms to their (unique) $\rightarrow_{pq}$-normal forms. As it turns out, $\downarrow_{pq}$ is implemented by ϕ.

Lemma 37 ([23]) $\forall j \in J^m \,.\, \phi(j) = \downarrow_{pq} (j)$

From here, we get the final core property of λJ^m.

Theorem 38 (Permutative Convertibility [23])

$$\forall j_1, j_2 \in J^m \,.\, j_1 =_{pq} j_2 \;\Leftrightarrow\; \phi(j_1) = \phi(j_2)$$

6.2.4 Diagram Equivalences and Consequences

From our perspective, Theorem 38 means that we collapse J^m(-terms) to $\mathcal{G}$(-terms) under permutative convertibility.

Lemma 39 $(Coll_{\mathcal{G}}^{J^m})_{pq} \wedge (StrColl_{\mathcal{G}}^{J^m})_{\hat{p}\hat{q}}$*, witnessed by* ϕ*.*

Remembering that $\mathcal{G} \subseteq J^m$, we note that we have the following.

Proposition 40 ([23]) $\forall g \in \mathcal{G} . \phi(g) = g$.

This (fairly straightforwardly) implies that we have some-reflection.

Lemma 41 $(sRefl_{\beta\mathcal{G}}^{\beta_1 X})$*, for any* $X \subseteq \{\beta_2, \pi, \mu\}$*, witnessed by* ϕ*.*

In order to get preservation, we note that permutative conversions can convert any J^m(-term) into a $\mathcal{G}$(-term) and that this either preserves or eliminates reduction steps.

Proposition 42 ([24])

- $\forall j_1, j_2 \in J^m \,.\, j_1 \rightarrow_{\beta_1\beta_2} j_2 \;\Rightarrow\; \phi(j_1) \twoheadrightarrow_{\beta\mathcal{G}} \phi(j_2)$
- $\forall j_1, j_2 \in J^m \,.\, j_1 \rightarrow_{\pi\mu} j_2 \;\Rightarrow\; \phi(j_1) = \phi(j_2)$

Lemma 43 $(Pres_{\beta\mathcal{G}}^{X}\text{-})$*, for any* $X \subseteq \{\beta_1, \beta_2, \pi, \mu\}$*, witnessed by* ϕ*.*

In other words, λJ^m and most of its sub-systems are diagram-equivalent with the λ-calculus.

Theorem 44 $\Delta\text{-}Eq(\twoheadrightarrow_{\beta_1 X \hat{p}\hat{q}}, \twoheadrightarrow_{\beta})$*, for* $X \subseteq \{\beta_2, \pi, \mu\}$.

Proof Theorem 30 and Lemma 8 applied to Lemma 39, Lemma 41 (via 21), and Lemma 43 (via 17). □

Among many other properties, we thus have the following.

Theorem 45 $\rightarrow_{\beta_1 X \hat{p}\hat{q}}$ *is confluent, for* $X \subseteq \{\beta_2, \pi, \mu\}$.

Proof The λ-calculus is confluent and confluence is expressible as a diagram, cf. Lemma 3. □

We note that is not clear whether or not we are able to state any confluence property for the λJ^m-calculus that does not involve $\hat{pq}$. More λJ^m-specific lemmas would probably be needed, although a more general version of our results might also suffice.

As a final remark, we note that [23, 22, 24] factor their work through two orthogonal, intermediary calculi. Our approach naturally applies there, too.

7 Extending the Language of Diagrams

The formalised notion of diagram presented in Section 2 is a simple, albeit expressive, one. In this section, we discuss a number of extensions to the formalisation, adding further to diagrams' expressivity, while retaining the ability to show diagram equivalence for relations that are not simply isomorphic. In other words, we seek diagram extensions that not only allow for a sensible definition of satisfiability with respect to a given relation, but which also retain the ability to show diagram equivalence by checking reflection and preservation properties of candidate homomorphisms.

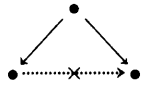
7.1 Negated Links

It is possible to add negated links to our diagrams. Graphically, we propose that these links be drawn with an $\times$ symbol super-imposed:

The above graph ("no terminal object") can be read

$$\forall y.\ \exists x.\ \neg R(x,y)$$

Another example is



stating that no divergence has an immediate resolution.

Where before we represented links in the "forall" and "existential" relations as triples of the form $(i, from, to)$, we extend these to 4-tuples: $(i, from, to, pos?)$, where the fourth component is a boolean value indicating whether or not the link is negated (true indicating that it is not negated, say). For example, the triangular diagram above would be formally represented by $\langle\{1,2,3\},\emptyset,F,E\rangle$ with

$$\begin{aligned} F &= \{(0,1,2,\top),(0,1,3,\top)\} \\ E &= \{(0,\mathsf{inl}(2),\mathsf{inl}(3),\bot)\} \end{aligned}$$

The evaluation of such diagrams with respect to a particular relation is of the same basic shape as Definition 2, but where the homomorphisms f and g of that definition now respect the extra boolean. In the case of f, for example:

$$\begin{aligned} (i,b_1,b_2,\top) \in F &\Rightarrow R_i(f(b_1),f(b_2)) \\ (i,b_1,b_2,\bot) \in F &\Rightarrow \neg R_i(f(b_1),f(b_2)) \end{aligned}$$

Lemma 46 (after Lemma 8)

$$(Pres^x_y) \wedge (aRefl^x_y) \Rightarrow \Delta\text{-}Eq(\rightarrow_x, \rightarrow_y)$$

where diagram-equivalence here means that the two relational structures are true for diagrams including negated links.

Proof If f is our homomorphism, then the conditions (Pres^x_y) and (aRefl^x_y) together ensure that

$$u \rightarrow_x v \Leftrightarrow f(u) \rightarrow_y f(v)$$

while onto-ness gives us the inverse that allows various homomorphisms to compose. The proof is identical to that of Lemma 8. □

7.2 (Reflexive and) Transitive Closure

The following diagram arises as a proof obligation when showing that the diamond property for R implies the diamond property for R^*.

To capture this within our language of diagrams, we might treat $\rightarrow$ and $\twoheadrightarrow$ as two independent relations in the diagram, and to then evaluate the diagram with respect to two relations, where the second was the reflexive and transitive closure of the first.

Unfortunately, if we wish to retain our preservation and reflection result, it is *not* possible to extend our diagrams so that the example above is evaluated with respect to just one relation, and where the $\twoheadrightarrow$ links must correspond to the reflexive and transitive closure of that relation because

$$u \rightarrow_x v \Leftrightarrow f(u) \rightarrow_y f(v)$$

does *not* imply

$$u \rightarrow^*_x v \Leftrightarrow f(u) \rightarrow^*_y f(v)$$

(see also Proposition 18).

By way of contrast, transitive closures *are* preserved and reflected by strong homomorphisms, allowing us to add transitive links to our language of diagrams. Finally, the counter-examples in Proposition 18 prompt one last attempt to derive a treatment of equality that is not just preserved but also reflected by strong homomorphisms.

7.3 Axiomatising Equality

If we extend our language so that diagrams can be combined in a propositional way, we can add extra constraints to our diagrams. Such constraints can then attempt to directly characterise equality. We begin by defining a propositional language of diagrammatic formulas:

$$DF \quad ::= \quad d \mid DF \wedge DF \mid \neg DF$$

with d a diagram as already defined. Just as with diagrams, the evaluation of such formulas is with respect to a family of relations.

Definition 47 *We write $R \models_f \phi$ to mean that diagrammatic formula ϕ is true of the family of relations R. This notion is simply defined:*

$$\begin{aligned} R \models_f d &= R \models d \\ R \models_f (\phi \wedge \psi) &= R \models_f \phi \text{ and } R \models_f \psi \\ R \models_f \neg\phi &= R \not\models_f \phi \end{aligned}$$

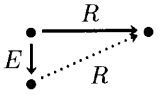
Lemma 48 (Any-Reflected DF-equivalence)

$$(Pres_a^c) \wedge (aReff_a^c) \Rightarrow$$

$$\forall \phi.\ \ \rightarrow_c \models_f \phi \ \Leftrightarrow\ \rightarrow_a \models_f \phi$$

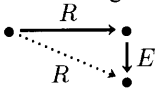
Proof By structural induction on the structure of ϕ, using Lemma 8 for the base case. □

We can now construct diagrams asserting that a relation is both an equivalence and a congruence with respect to the other relations we are concerned with. Congruence on the left is the diagram



(Here the labelling does not mean that the diagram is being evaluated with respect to a particular pair of relations. The labelling simply distinguishes the two different relations.)

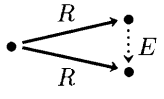
Similarly, congruence on the right is



Diagrams asserting that E is reflexive, symmetric and transitive are easy to construct. Defining $\phi \Rightarrow \psi$ on diagrammatic formulas as $\neg(\phi \wedge \neg\psi)$, we can now express the determinism formula as

$$\begin{array}{l} \mathit{reflexive}(E) \wedge \mathit{symmetric}(E) \wedge \\ \mathit{transitive}(E) \wedge \mathit{congruent}(R, E) \end{array}$$

$\Rightarrow$



Thanks to Lemma 48, we know that this formula *will* be preserved and reflected. Unfortunately, we can not be sure that we will have normal models for the diagrams. In other words, the constraints do not require E to be the identity, only that it be an equivalence relation that "respects" the various reduction relations we are interested in.

8 Conclusion

We have formalised a language of diagrams of the sort that is widely used informally to express behavioural properties of rewrite relations. The language gives a simple semantics for a variety of Π_1-formulas, including negated relationships. We have used this to give an algebraic proof that two rewrite relations that are related by a *structural collapse* enjoy all the same properties, expressed as diagrams. The development includes results that lower the threshold for applying the core lemma to the point where *diagram equivalence* holds by construction when, e.g., doing an α-collapse in the Hindley-Curry sense. The paper is partly justified by the large number of recently-proposed formalisms for reasoning about languages with binding, and we have shown that at least four different types for the λ-calculus are diagram equivalent. A completely differently-flavoured example shows diagram equivalence between β-reduction and cut-elimination, modulo permutative conversions.

Acknowledgements We would like to thank José Espírito Santo and Luis Pinto for discussions about the material covered in Section 6.2 and for making [24] available to us. National ICT Australia is funded by the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

Availability The HOL4 sources mechanising the theory of diagrams (extended with negated and transitive closure links), diagrammatic formulas (from Section 7.3) and the λ-calculus application of Section 6.1 are available from `http://hol.cvs.sourceforge.net/hol/hol98/examples/lambda/`. The same sources will also be part of the next release of the HOL system.

References

[1] Simon J. Ambler, Roy L. Crole, and Alberto Momigliano. A definitional approach to primitive recursion over higher order abstract syntax. In F. Honsell, M. Miculan, and A. Momigliano, editors, *Proceedings of the ACM SIGPLAN Workshop: MERLIN-2*. ACM Digital Library, 2003. `http://doi.acm.org/10.1145/976571.976572`.

[2] Brian E. Aydemir, Aaron Bohannon, Matthew Fairbairn, J. Nathan Foster, Benjamin C. Pierce, Peter Sewell, Dimitrios Vytiniotis, Geoffrey Washburn, Stephanie Weirich, and Steve Zdancewic. Mechanized metatheory for the masses: the POPLMARK challenge. In Hurd and Melham [13].

[3] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[4] H. P. Barendregt. *The Lambda Calculus: its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, Amsterdam, revised edition, 1984.

[5] Serenella Cerrito and Delia Kesner. Pattern matching as cut elimination. In Giuseppe Longo, editor, *Proceedings of LICS-14*, pages 98–108. IEEE CS Press, 1999.

[6] Roy Dyckhoff and Luis Pinto. Permutability of proofs in intuitionistic sequent calculi. *Theoretical Computer Science*, 212(1–2):141–155, 1999.

[7] Gerhard Gentzen. Investigations into logical deduction. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*. North-Holland, 1969.

[8] R. I. Goldblatt and S. K. Thomason. Axiomatic classes in propositional modal logic. In J. Crossley, editor, *Algebra and Logic*, pages 163–173. 1974, 1974.

[9] A. D. Gordon and T. Melham. Five axioms of alpha conversion. In J. von Wright, J. Grundy, and J. Harrison, editors, *Theorem Proving in Higher Order Logics: 9th International Conference, TPHOLs'96*, volume 1125 of *Lecture Notes in Computer Science*, pages 173–190. Springer-Verlag, 1996.

[10] H. Herbelin. A λ-calculus structure isomorphic to Gentzen-style sequent calculus structure. In *Annual Conference of the European Association for Computer Science Logic, CSL'94, Kazimierz (Poland), Selected Papers*, volume 933 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[11] Peter Homeier. A proof of the Church-Rosser theorem for the lambda calculus in higher order logic. In Richard J. Boulton and Paul B. Jackson, editors, *TPHOLs 2001: Supplemental Proceedings*, pages 207–222. Division of Informatics, University of Edinburgh, September 2001. Available as Informatics Research Report EDI-INF-RR-0046.

[12] W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism*, pages 470–490. Academic Press, 1980.

[13] Joe Hurd and T. Melham, editors. *Theorem Proving in Higher Order Logics, 18th International Conference*, volume 3603 of *LNCS*. Springer, 2005.

[14] Stephen C. Kleene. Permutability of inferences in Gentzen's calculi LK and LJ. *Memoirs of the American Mathematical Society*, 10:1–26, 1952.

[15] Georg Kreisel. A survey of proof theory II. In J. E. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*, pages 109–170. North-Holland, 1971.

[16] Michael Norrish. Recursive function definition for types with binders. In Konrad Slind, Annette Bunker, and Ganesh Gopalakrishnan, editors, *Theorem Proving in Higher Order Logics, 17th International Conference*, volume 3223 of *LNCS*, pages 241–256. Springer, September 2004.

[17] Michael Norrish. Mechanising λ-calculus using a classical first order theory of terms with permutations. *Higher Order and Symbolic Computation*, To appear.

[18] A. M. Pitts. Alpha-structural recursion and induction (extended abstract). In Hurd and Melham [13], pages 17–34.

[19] Garrel Pottinger. Normalization as a homomorphic image of cut-elimination. *Annals of Mathematical Logic*, 12:323–357, 1977. North-Holland Publishing Company.

[20] Dag Prawitz. *Natural Deduction*. Almquist and Wiksell, Uppsala, 1965.

[21] José Espírito Santo. *Conservative Extensions of the lambda-Calculus for the computational interpretation of sequent calculus*. PhD thesis, University of Edinburgh, 2002.

[22] José Espírito Santo and Luis Pinto. Confluence and strong normalisation of the generalised multiary lambda-calculus. In *Proceedings of the 3rd International Workshop on Types for Proofs and Programs, TYPES*, pages 194–209, 2003.

[23] José Espírito Santo and Luis Pinto. Permutative conversions in intuitionistic multiary sequent calculi with cuts. In *Proceedings of TLCA-6*, pages 286–300, 2003.

[24] José Espírito Santo and Luis Pinto. A calculus of multiary sequent terms. In preparation, 2006.

[25] Helmut Schwichtenberg. Termination of permutative conversions in intuitionistic Gentzen calculi. *Theoretical Computer Science*, 212(1–2):247–260, 1999.

[26] René Vestergaard. *The Primitive Proof Theory of the λ-Calculus*. PhD thesis, School of Mathematical and Computer Sciences, Heriot-Watt University, 2003.

[27] René Vestergaard and James Brotherston. A formalised first-order confluence proof for the λ-calculus using one-sorted variable names. *Information and Computation*, 183(2):212 – 244, 2003. Special edition with selected papers from RTA01.

[28] René Vestergaard and Joe Wells. Cut rules and explicit substitutions. *Mathematical Structures in Computer Science*, 11(1):131–168, 2001. Special issue on explicit substitutions, with selected papers from WESTAPP'99.

[29] J. von Plato. Natural deduction with general elimination rules. *Arch. for Mathematical Logic*, 40(7):541–567, 2001.

[30] J. Zucker. The correspondence between cut-elimination and normalization, Part I & II. *Annals of Mathematical Logic*, 7:1–112,113–155, 1974. North-Holland.