Electronic Notes in Theoretical Computer Science 70 No. 6 (2002)
URL: `http://www.elsevier.nl/locate/entcs/volume70.html` 41 pages

Four equivalent equivalences of reductions

Vincent van Oostrom [1]

Department of Philosophy
Universiteit Utrecht
Utrecht, The Netherlands

Roel de Vrijer [2]

Department of Theoretical Computer Science
Vrije Universiteit
Amsterdam, The Netherlands

Abstract

Two co-initial reductions in a term rewriting system are said to be equivalent if they perform the same steps, albeit maybe in a different order. We present four characterisations of such a notion of equivalence, based on permutation, standardisation, labelling and projection, respectively. We prove that the characterisations all yield the same notion of equivalence, for the class of first-order left-linear term rewriting systems. A crucial rôle in our development is played by the notion of a proof term.

1 Introduction

We consider reductions up to the order in which orthogonal steps are performed. The heart of the paper consists of the following four characterisations of the ensuing notion of equivalence of reductions. Two reductions between the same two terms are

(i) *permutation* equivalent if they can be transformed into each other by repeated permutation of orthogonal steps (Section 3),

(ii) *parallel standardisation* equivalent if sorting the steps into outside-in order yields the same parallel standard reduction (Section 4),

(iii) *labelling* equivalent if their targets are the same for any labelling of their sources (Section 5),

[1] Email: `Vincent.vanOostrom@phil.uu.nl`
[2] Email: `rdv@cs.vu.nl`

(iv) *projection* equivalent if the projection of either reduction over the other results in the empty reduction (Section 6).

In fact, these equivalence are most naturally defined on *proof terms* and we will do so. Proof terms represent proofs in Meseguer's rewriting logic [36] as terms. Hence they can directly represent both sequential computations (reductions) as well as parallel ones (see e.g. Example 2.12). All four characterisations are equivalent, as will be pointed out in Section 7, which allows one to transfer results obtained for one to the others. This is useful, since each characterisation stresses a particular view on the equivalence (permutation, ordering, labelling, projection) and comes with its own application area and results. The different notions of equivalence have been studied before, but mainly in the context of orthogonal rewriting. As far as we know, this is the first systematic study relating these notions in the context of left-linear term rewrite systems, possibly having critical pairs.

This paper can be seen as an overview of part of [47] Chapter 8, where all real proofs can be found. The proofs provided in this paper are in fact only proof sketches. We will just try to motivate and illustrate our characterisations. To that end, we employ the following running example. [3]

Example 1.1 Let $\mathcal{U}$ be the TRS with the three rules

$$\begin{aligned} a &\to b \\ f(x,b) &\to g(x) \\ g(b) &\to c \end{aligned}$$

and consider the following two reductions from $f(a,a)$ to the term c.

$$f(\tilde{a},a) \to f(b,\overline{a}) \to \underline{f}(b,\underline{b}) \to \underline{g}(\underline{b}) \to c$$
$$f(a,\overline{a}) \to \underline{f}(a,\underline{b}) \to g(\tilde{a}) \to \underline{g}(\underline{b}) \to c$$

Intuitively, these reductions perform the same steps, but in a different order. Indeed, we will show them equivalent for each characterisation.

Remark 1.2 The common features of all of these notions of equivalence can also be captured in a syntax free way by means of event structures ([49]). However, note that this only holds in the case of linear, i.e. non-erasing and non-duplicating, events. In particular, the obvious idea of modelling term rewrite steps as events fails, since term rewrite steps might be erasing (K-steps in combinatory logic) or duplicating (S-steps). In fact, it is exactly the replicating, i.e. erasing and duplicating, behaviour of term rewrite steps that makes the results in this paper non-trivial. It is an area of active research how to generalise event structures in order to deal adequately with erasure and duplication. For instance, In [20] an event-style semantics for conflict free rewrite systems is presented, including first-order orthogonal term rewrite

[3] For ease of exposition we have chosen a very well-behaved TRS as running example; it is linear orthogonal in the sense of [41], hence it is uniformly normalising.

systems and the $\lambda\beta$-calculus, among others. Since it is not clear yet whether, and if so how, this can be generalised to rewrite systems with conflicts, and since we are interested in non-orthogonal systems as well, we do not treat event structures in this paper.

In a series of papers ([34,33,31,32,35]) Melliès has set out to axiomatise rewriting theory in a syntax free way, his training partner being the $\lambda\sigma$-calculus. His axioms apply to the systems we consider. Therefore several of the results and methods presented here, in particular those on parallel standardisation, have a syntax free analogue in his theory.

Acknowledgement

We gratefully acknowledge the support of the following people when writing [47] Chapter 8: Eduardo Bonelli, Sander Bruggink, Zurab Khasidashvili, Jan Willem Klop, Paul-André Melliès, Aart Middeldorp, Paula Severi, Yde Venema, René Vestergaard, Albert Visser, and Fer-Jan de Vries. We thank the organisers of the WRS and HOR workshops affiliated to the RTA '02 for giving us the opportunity to present our work, and the organisers of the former for inviting us to contribute to this special issue. The first author would like to thank the university of Tsukuba, ETL in Tsukuba and CWI in Amsterdam, for providing hospitality and funding.

2 Proof terms

As explained above, we are interested in reductions up to the order of steps in them. Obviously, steps can only be sensibly reordered if there is a notion of identity on them. For that purpose associating a *rewrite relation* to a term rewrite system is inadequate because of so-called *syntactic accidents* ([24]).

Example 2.1 Consider the TRS $\mathcal{T}$ with the single rule $f(x) \to x$. There are two steps from $f(f(a))$, indicated by underlining the head symbols of their redex-patterns:

$$\underline{f}(f(a)) \to f(a) \qquad f(\underline{f}(a)) \to f(a)$$

Note that these steps would cause a syntactic accident: both steps give rise to the step $f(f(a)) \to_{\mathcal{T}} f(a)$ in the underlying rewrite relation $\to_{\mathcal{T}}$ of $\mathcal{T}$.

To overcome the difficulty we will instead associate an *abstract rewrite system* (see Remark 2.5 for our use of the notion of abstract rewrite system) to a term rewrite system. The idea is to have an explicit witness to the way in which a term reduces to another term, for a given term rewrite system. Such witnesses will be called *proof terms.*

2.1 Abstract rewrite systems

Definition 2.2 An *abstract rewrite system*, ARS for short, is a quadruple $\langle A, \Phi, \mathsf{src}, \mathsf{tgt}\rangle$ with A a set of *objects*, Φ a set of *steps* and src, $\mathsf{tgt} : \Phi \to A$ the *source* and *target* functions, respectively (see Figure 1).

We will employ various arrow-like symbols $\to$, $\leadsto$, $\Rightarrow$, ... to range over ARSs, employ a, b, c, ... to range over objects, and ϕ, ψ, χ, ... to range over steps. For a given abstract rewrite system $\to$, we write either $\phi : a \to b$ (in text), or $a \to_\phi b$ (in formulas), or even $a\ \phi\ b$ to indicate that ϕ is a step with source a and target b; ϕ is a *witness* to the claim that some step from a to b exists.

$$\begin{tikzcd}
\text{object} & \text{step} \arrow[l, "\mathsf{src}"'] \arrow[r, "\mathsf{tgt}"] & \text{object} \\
\text{term} & \text{rule} \arrow[l, "\mathsf{lhs}"'] \arrow[r, "\mathsf{rhs}"] & \text{term}
\end{tikzcd}$$

Fig. 1. Abstract rewrite system (top) and Rule rewrite system (bottom)

Example 2.3 (i) The *black hole* or *loop* ARS has a single object $\bullet$ and a single step from the object to itself.

(ii) For every natural number n, the ARS $\to^n$ has objects $\bullet_1, \ldots, \bullet_n$ and steps $i+1 : \bullet_i \to^n \bullet_{i+1}$, for every i such that $1 \le i$ and $i+1 \le n$.

(iii) The ARS $-\!\!\infty\!\!\rightarrow$ is the union of $\to^n$ for all natural numbers n. That is, $-\!\!\infty\!\!\rightarrow$ is just the infinite 'straight line' $\bullet_1 \to^1 \bullet_2 \to^2 \bullet_3 \to^3 \cdots$.

(iv) The *syntactic accident* ARS $\rightrightarrows$ consists of two objects and two steps, in the same direction, between them.

Steps having the same source will be called *co-initial* and steps having the same target *cofinal*. A step ψ is *composable* with a step ϕ, if the source of ψ is the same as the target of ϕ. A *loop* is a step the source and target of which coincide. The strongest properties one usually shows for multi-steps (see below) in an orthogonal term rewrite system, are the diamond and the triangle properties.

Definition 2.4 (i) $\to$ has the *diamond* property, if for any co-initial steps ϕ_1, ϕ_2, there exist cofinal steps ψ_1, ψ_2, such that ψ_i is composable with ϕ_i.

(ii) $\to$ has the *triangle* property, if for any object a, there exists an object a^* such that for any step ϕ from a, there is a step composable with it to a^*.

Note that if an ARS has the triangle property, then it has the diamond property. Moreover, in case it also has all loops, $a \to a^*$ holds for every object a (see Figure 2).

$$\begin{tikzcd}
& a \arrow[dl] \arrow[dr] & \\
b \arrow[dr, dashed] & & c \arrow[dl, dashed] \\
& d &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
a \arrow[dr] \arrow[dd, dashed] & \\
& b \arrow[dl, dashed] \\
a^* &
\end{tikzcd}$$

Fig. 2. The diamond property and (for ARSs having all loops) the triangle property

Remark 2.5 Definition 2.2 is in concordance with many papers on abstract rewriting in general and residuals in particular: [38,44,14,45,30]. The terminology used in connexion with abstract rewrite systems varies throughout the literature, depending on the intended application area. For instance, our abstract rewrite systems are called *indexed* 1*-complexes* in [38], *reduction complexes* in [44], and *graphs* in [25]. Objects are called *points*, *vertices*, *states* or *worlds*. Steps are called *(*1*-)cells*, *transitions*, *moves*, *events*, *edges* or *arrows*. Source and target are called *domain* and *codomain*, *start* and *terminus*, or *initial* and *final*. We warn the reader that in the literature ARSs may denote either the ARSs as employed here (typically by the authors of the papers mentioned above and in Chapters 8 and 9 of [47]), or rewrite relations (typically in [22] and in the other chapters of [47]). For this reason, we avoid using (rewrite) relational terminology for our ARSs, which is in concordance with [38]:

> The notions that arise are closely related to those of the theory of partially ordered sets, but usually not identical. Except in the case of identity the terms of that theory are therefore avoided.

2.2 Term rewrite systems

Definition 2.6 A *term rewrite system*, TRS for short, $\mathcal{T}$ is a structure $\langle \Sigma, R\rangle$ such that Σ is a signature and R is an ARS having terms over (variables and) Σ as objects.

The steps of R are called *rules* and the source and target of a rule are called its *left-hand side* (lhs) and *right-hand side* (rhs), respectively (see Figure 1). We employ ϱ, ϑ, ς, ... to range over rules. We require for a rule $\varrho : l \to r$:

(i) l is not a variable,

(ii) the variables in r are among those in l

We will moreover require *left-linearity*, i.e. variables occur at most once in l.

Example 2.7 The signature of our running example TRS $\mathcal{U}$ is $\{a, b, c, f, g\}$, with arities 0, 0, 0, 2 and 1, respectively. We name its, left-linear, rules:

$$\begin{aligned} \varrho &: a \to b \\ \vartheta &: f(x,b) \to g(x) \\ \varsigma &: g(b) \to c \end{aligned}$$

For a given TRS, its *proof terms* are terms over the signature for terms, extended with symbols representing the various inference rules of equational logic. The *rules* of the TRS are adjoined to the signature as so-called *rule* symbols.

Definition 2.8 Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a term rewrite system. To every rule ϱ in R, a *rule* symbol, also denoted by ϱ, is associated having the number of free variables in its left-hand side as arity. The *proof term* signature is the (disjoint) union of Σ, the set of rule symbols of R, and $\{\cdot\}$, where $\cdot$ is the binary *composition* symbol. The *underlying* abstract rewrite system $\geq_{\mathcal{T}}$ of $\mathcal{T}$, $\geq$ for short, is defined as follows.

- The objects are terms over Σ.
- The steps are called *proof terms* or just *proofs* and are an inductively defined subset of the terms over the proof term signature. We employ ϕ, ψ, χ, ... to range over proof terms. Steps of $\geq$ are defined together with their source and targets by the following inference system.

$$\frac{\phi_1 : s_1 \geq t_1 \quad \ldots \quad \phi_n : s_n \geq t_n}{\varrho(\phi_1,\ldots,\phi_n) : l(s_1,\ldots,s_n) \geq r(t_1,\ldots,t_n)}\ \text{(rule)}$$

$$\frac{\phi_1 : s_1 \geq t_1 \quad \ldots \quad \phi_n : s_n \geq t_n}{f(\phi_1,\ldots,\phi_n) : f(s_1,\ldots,s_n) \geq f(t_1,\ldots,t_n)}\ \text{(replacement)}$$

$$\frac{\phi : s \geq t \quad \psi : t \geq u}{(\phi \cdot \psi) : s \geq u}\ \text{(transitivity)}$$

In the (rule)-rule ϱ is a rule symbol such that its corresponding rule $\varrho : l \to r$ in R contains n variables. In the (replacement)-rule f is an n-ary function symbol in Σ.

Note that the objects of the proof term ARS $\geq$ are (unlike the lhs and rhs of the rules of the TRS) closed terms, i.e. they do not contain variables. For a justification of this restriction, see Remark 2.15. In the (rule)-rule, we have employed the following notational convention.

Notation 2.9 *In case $\varrho : l \to r$ is a rule, $l(s_1,\ldots,s_n)$ and $r(s_1,\ldots,s_n)$ denote the terms obtained by substituting s_i in l and r, respectively, for the ith variable of ϱ. Here we assume variables to be ordered in some arbitrary but fixed way depending on ϱ. The other way around, we will also employ ϱ^σ to denote the proof term $\varrho(x_1^\sigma,\ldots,x_n^\sigma)$, if σ is a substitution for the variables in ϱ.*

Using proof terms, syntactic accidents disappear.[4]

Example 2.10 Let $\varrho : f(x) \to x$ be the named version of the rule of the syntactic accident TRS in Example 2.1. Then, $\varrho(f(a))$ and $f(\varrho(a))$ both witness $f(f(a)) \geq f(a)$. In the former case this conclusion is reached by:

$$\dfrac{\dfrac{\dfrac{}{a : a \geq a}\,(\text{replacement})}{f(a) : f(a) \geq f(a)}\,(\text{replacement})}{\varrho(f(a)) : f(f(a)) \geq f(a)}\,(\text{rule})$$

In the latter case it is inferred from

$$\dfrac{\dfrac{\dfrac{}{a : a \geq a}\,(\text{replacement})}{\varrho(a) : f(a) \geq a}\,(\text{rule})}{f(\varrho(a)) : f(f(a)) \geq f(a)}\,(\text{replacement})$$

The usual ARSs associated to a TRS arise by restricting restricting proof terms in appropriate ways (see Figure 3).

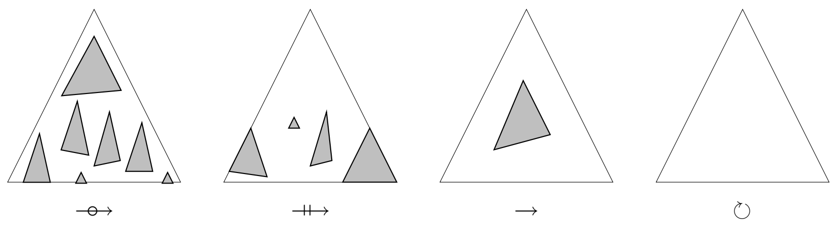
Fig. 3. Multi-step, parallel step, step, and loop

Definition 2.11 A proof term constructed without using the (transitivity)-rule, i.e. without occurrences of $\cdot$, is called:

(i) a *multi*-step (and the restriction of $\geq$ to multi-steps is) denoted by $\circ\!\!\!\!\longrightarrow$.

(ii) a *parallel step*, denoted by $\Vert\!\!\!\!\longrightarrow$, if rule symbols do not occur nested.

(iii) a (*one-*)*step*, denoted by $\to$, if exactly one rule symbol occurs in it.

(iv) a *loop*, denoted by $\circlearrowleft$, if no rule symbols occur in it.

A multi-step is a step simultaneously performing an arbitrary number of orthogonal ordinary steps. A parallel step is a special kind of multi-step, where the redex-patterns involved occur at parallel positions, they are disjoint. Multi-steps and parallel steps are interesting since, roughly speaking, they are the least extensions of the steps of an orthogonal TRS, which satisfy the diamond and triangle properties, respectively. Steps themselves usually do

[4] Even stronger: only now can we *express* what it means for a TRS $\mathcal{T}$ to *have* syntactic accidents: the syntactic accident ARS of Example 2.3 must be embeddable into its one-step ARS $\to_{\mathcal{T}}$ (see Definition 2.11).

not satisfy either of these properties due to possible non-right-linearity of term rewrite rules.

Example 2.12 (i) Consider the TRS $\{\varrho : a \to b, \vartheta : f(x) \to g(x,x)\}$. The co-initial steps $f(\varrho) : f(a) \to f(b)$ and $\vartheta(a) : f(a) \to g(a,a)$ cannot be made into a diamond by any pair of cofinal steps. However, viewing them as parallel steps, they can be made into a diamond by means of $\vartheta(b) : f(b) \mathrel{-\!\!\!\shortparallel\!\!\!\to} g(b,b)$ and $g(\varrho,\varrho) : g(a,a) \mathrel{-\!\!\!\shortparallel\!\!\!\to} g(b,b)$; $\mathrel{-\!\!\!\shortparallel\!\!\!\to}$ has the diamond property. Still, parallel steps do not have the triangle property since e.g. the common reduct $g(b,b)$ cannot be reached in one parallel step from $f(a)$. However, it *can* be reached using one multi-step: $\vartheta(\varrho) : f(a) \mathrel{-\!\!\circ\!\!\!\to} g(b,b)$; $\mathrel{-\!\!\circ\!\!\!\to}$ has the triangle property.

(ii) Consider the TRS $\{\varrho : a \to a\}$. The step $a \geq a$ is witnessed both by the one-step $\varrho : a \to a$, via an application of (rule) for ϱ (note that ϱ is nullary), and by the loop $a : a \circlearrowright a$, by an application of (replacement).

From the second example it should be clear that what were defined to be loop steps in Definition 2.11, are indeed loops for the ARS $\geq$, in the sense that they have identical sources and targets. The example also shows that the ARS $\geq$ might have other loops in this sense as well, for instance those generated by loop rules (ϱ in the example).

The reductions in Example 1.1 can be seen as special proof terms as well.

Definition 2.13 A proof term is a *reduction*, denoted by $\twoheadrightarrow$, if it is either a loop or constructed using (transitivity) from steps, modulo the reduction identities, i.e. the first three identities in Table 1 on page 10

Example 2.14 Consider the 'reductions' of Example 1.1 again:

$$f(\tilde{a},a) \to f(b,\overline{a}) \to \underline{f}(b,\underline{b}) \to \underline{g}(\underline{b}) \to c$$
$$f(a,\overline{a}) \to \underline{f}(a,\underline{b}) \to g(\tilde{a}) \to \underline{g}(\underline{b}) \to c$$

They can be witnessed by reductions using the proof term signature of Example 2.7:

$$f(\varrho,a) \cdot f(b,\varrho) \cdot \vartheta(b) \cdot \varsigma : f(a,a) \twoheadrightarrow c$$
$$f(a,\varrho) \cdot \vartheta(a) \cdot g(\varrho) \cdot \varsigma : f(a,a) \twoheadrightarrow c$$

Below, we will abbreviate these two proof terms to $\mathcal{R}$ and $\mathcal{S}$, respectively.

Remark 2.15 (i) Usually the rules of a term rewrite system are just pairs of terms. Consequently, one usually is fuzzy about the identity of rules. For instance, the rules $f(x) \to g(x)$ and $f(y) \to g(y)$ formally are distinct as pairs of terms, but they are nevertheless often identified, since they are *variants* of each other. This problem is easily overcome in the present setting, by defining an appropriate quotient on rules.

(ii) Note that we *do not* employ proof terms for the whole of equational logic. Notably, we do not employ the following proof terms corresponding to the inference rules for reflexivity and symmetry:

$$\frac{}{\circlearrowleft_s : s \geq s}\,\text{(reflexivity)} \qquad \frac{\phi : s \geq t}{\phi^{-1} : t \geq s}\,\text{(symmetry)}$$

Conceptually, it would be better to have the inference rule for reflexivity and indeed it is present in rewriting logic. However for our present purposes, studying permutation equivalence, it is technically redundant. It might seem problematic that the (replacement)-rule only applies to function symbols, not to variables, so $s : s \geq s$ holds only for s closed. However, variables behave as constants with respect to permutation equivalence. Since here we are interested in permutation equivalence, we may simply assume that reductions/proof terms are closed.

As in rewriting logic, the inference rule for symmetry is absent since we are only interested in permutation equivalence of *reductions*, not of *conversions*.

(iii) Making reductions or, more generally, proofs explicit, by turning them into terms and manipulating these, is a natural idea as it is an instance of the general idea of explicitly representing a concept in order to prove properties about it. For instance, this general idea is at the basis of studying substitution via explicit substitution calculi. Our proof terms represent proofs in rewriting logic ([36]), i.e. equational logic without symmetry. An outline to this active research area can be found in [28]. A case study in the proof theory of rewriting for the simply typed λ-calculus with β-reduction and η-expansion (in its De Bruijn representation), having similar aims as the present paper, has been carried out in [13]. Related concepts and related notions of proof terms occur at many places in the literature, see e.g. [46,7,10,17].

(iv) Note that multi-steps can be *developed* into reductions by applying the rules in them one by one. For instance, $\vartheta(\varrho)$ in Example 2.12 can be developed into each of $\vartheta(a)\cdot g(a,\varrho)\cdot g(\varrho,b)$, $\vartheta(a)\cdot g(\varrho,a)\cdot g(b,\varrho)$, and $f(\varrho)\cdot\vartheta(b)$. In fact, all developments of a multi-step are finite and have the same target, which is known as the *finite developments* theorem.

3 Permutation equivalence

We introduce our first notion of equivalence, *permutation* equivalence. Two reductions are permutation equivalent if they perform the same steps but possibly in a different order. This is analogous to saying that the lists $[2,1,0]$ and $[0,1,2]$ have the same members, listed in a different order. In the case of these lists, this can be shown by repeatedly permuting pairs of adjacent members:

$$[2,\underline{1,0}] \cong [\underline{2,0},1] \cong [0,\underline{2,1}] \cong [0,1,2]$$

Similarly, two reductions will be said to be permutation equivalent if one can be obtained from the other by a series of permutations of adjacent steps. Actually, permutations will be defined on proof terms rather than on reductions. The idea is that this allows one to refine a single permutation into two serialisation steps. It is analogous to saying that the lists $[1,2]$ and $[2,1]$ are permutation equivalent since

$$[2,1] \cong [{}^{1}_{2}] \cong [1,2]$$

where the left and right lists are two distinct ways to serialise the middle one where both its members are 'concurrent'. Instead of stacking 1 and 2 on top of each other, to express their concurrency we could have written them next to each other, 1 2, as well. For lists this is just a matter of notation, but in proof terms there really are two distinct ways in which steps may be concurrent: horizontally and vertically. The definition of permutation equivalence is composed accordingly.

Example 3.1 The proof terms $\mathcal{R} = f(\varrho, a) \cdot f(b, \varrho) \cdot \vartheta(b) \cdot \varsigma$ and $\mathcal{S} = f(a, \varrho) \cdot \vartheta(a) \cdot g(\varrho) \cdot \varsigma$ of our running example, witnessing the reductions

$$f(\tilde{a}, a) \to f(b, \overline{a}) \to \underline{f}(b, \underline{b}) \to \underline{g}(\underline{b}) \to c$$
$$f(a, \overline{a}) \to \underline{f}(a, \underline{b}) \to g(\tilde{a}) \to \underline{g}(\underline{b}) \to c$$

can be shown to be permutation equivalent as follows:

$$\begin{aligned}
\mathcal{R} &= \underline{f(\varrho, a) \cdot f(b, \varrho)} \cdot \vartheta(b) \cdot \varsigma \\
&\cong f(\underline{\varrho \cdot b}, \underline{a \cdot \varrho}) \cdot \vartheta(b) \cdot \varsigma \\
&\cong f(\underline{\varrho}, \underline{\varrho}) \cdot \vartheta(b) \cdot \varsigma \\
&\cong \underline{f(a \cdot \varrho, \varrho \cdot b)} \cdot \vartheta(b) \cdot \varsigma \\
&\cong f(a, \varrho) \cdot \underline{f(\varrho, b) \cdot \vartheta(b)} \cdot \varsigma \\
&\cong f(a, \varrho) \cdot \underline{\vartheta(\varrho)} \cdot \varsigma \\
&\cong f(a, \varrho) \cdot \vartheta(a) \cdot g(\varrho) \cdot \varsigma \\
&= \mathcal{S}
\end{aligned}$$

where we have employed *all* the permutation identities of Table 1 below.

$$\boxed{\begin{aligned}
\circlearrowright \cdot \phi &\approx \phi \\
\phi \cdot \circlearrowright &\approx \phi \\
(\phi \cdot \psi) \cdot \chi &\approx \phi \cdot (\psi \cdot \chi) \\
f(\phi_1, \ldots, \phi_n) \cdot f(\psi_1, \ldots, \psi_n) &\approx f(\phi_1 \cdot \psi_1, \ldots, \phi_n \cdot \psi_n) \\
\varrho(\phi_1, \ldots, \phi_n) &\approx l(\phi_1, \ldots, \phi_n) \cdot \varrho(t_1, \ldots, t_n) \\
\varrho(\phi_1, \ldots, \phi_n) &\approx \varrho(s_1, \ldots, s_n) \cdot r(\phi_1, \ldots, \phi_n)
\end{aligned}}$$

Table 1
Permutation identities

Definition 3.2 (i) The first three identitities in Table 1 are the *reduction* identities.

(ii) The reduction identities combine with the fourth *functorial* identity, to form the *structural* identities. Actually the functorial identity is a schema: it is assumed for every function symbol f.

(iii) The structural identities combine with the fifth *inner* and sixth *outer* identity (schemata), for every rule symbol $\varrho : l \to r$, with $\phi_i : s_i \geq t_i$, for all $1 \leq i \leq n$, to form the *permutation* identities.

The generated *structural* and *permutation* equivalences on proof terms will be denoted by $\equiv$ and $\cong$, respectively. The *permutation* order $\sqsubseteq$ is defined by $\phi \sqsubseteq \chi$ if there exists ψ such that $\phi \cdot \psi \cong \chi$.

Note that permutation equivalence $\cong$ is contained in the permutation order $\sqsubseteq$, since $\phi \cdot \circlearrowleft \cong \phi$. Functorial identity captures horizontal concurrency and the inner and outer identities capture vertical concurrency.

Example 3.3 (i) Consider a TRS having a single rule $\varrho{:}\, a \to b$ and a binary function symbol g. The proof terms $g(\varrho, a) \cdot g(b, \varrho)$ and $g(a, \varrho) \cdot g(\varrho, b)$ both witness $g(a, a) \geq g(b, b)$. Either proof term performs both a-steps, but the former performs the left a-step first and the latter performs the right a-step first. They are structurally equivalent, since the a-redexes are parallel to one another, so could have been performed simultaneously:

$$\begin{aligned} \underline{g(\varrho, a) \cdot g(b, \varrho)} &\equiv g(\underline{\varrho \cdot b}, \underline{a \cdot \varrho}) \\ &\equiv g(\underline{\varrho}, \underline{\varrho}) \\ &\equiv \underline{g(a \cdot \varrho, \varrho \cdot b)} \\ &\equiv g(a, \varrho) \cdot g(\varrho, b) \end{aligned}$$

Observe that the middle proof term $g(\varrho, \varrho)$ is a parallel step witnessing the simultaneous application of two parallel occurrences of ϱ (corresponding to 1 2 in the discussion above).

(ii) Consider the TRS with rules $\varrho : a \to b$ and $\vartheta : f(x) \to g(x, x)$. The proof terms $f(\varrho) \cdot \vartheta(b)$ and $\vartheta(a) \cdot g(\varrho, \varrho)$ both witness $f(a) \geq g(b, b)$ and contract the g- and a-redexes present in the source. The former contracts the inner and the latter the outer redex first. Hence intuitively they perform the same steps. Their permutation equivalence can be read off either pictorially from Figure 4 or algebraically from $f(\varrho) \cdot \vartheta(b) \cong \vartheta(\varrho) \cong \vartheta(a) \cdot g(\varrho, \varrho)$. Note that the mediating proof term $\vartheta(\varrho)$ is a multistep witnessing the simultaneous application of the nested occurrences of ϱ and ϑ.

The second example shows that permuting nested steps (vertical permutation) is complex since it leads to non-linear phenomena if the outermost rule is non-right-linear (in the example: ϑ is duplicating). (Note that our running example above exhibits both horizontal and vertical permutations.) This is

$$\begin{tikzcd}
& & f(a) \arrow[lld, "f(\varrho)"'] \arrow[d, dash] \arrow[rrd, "\vartheta(a)"] & & \\
f(b) \arrow[rrd, "\vartheta(b)"'] & \cong & \vartheta(\varrho) \arrow[d] & \cong & g(a,a) \arrow[lld, "g(\varrho,\varrho)", "\shortparallel" marking] \\
& & g(b,b) & &
\end{tikzcd}$$

Fig. 4. Permutation equivalence of $f(\varrho) \cdot \vartheta(b)$ and $\vartheta(a) \cdot g(\varrho, \varrho)$.

what makes permutation equivalence complex. Nonetheless, the following results are easy, as they follow directly from the properties of equational logic.

Theorem 3.4 (i) *Permutation equivalence $\cong$ is a congruence.*

(ii) *The permutation order $\sqsubseteq$ is a quasi-order.*

(iii) *If $\phi \cong \psi$, then $\mathsf{src}(\phi) = \mathsf{src}(\psi)$ and $\mathsf{tgt}(\phi) = \mathsf{tgt}(\psi)$, and similarly for reductions (i.e. modulo the reduction identities).*

Remark 3.5 Permutation equivalence is of a relatively recent origin. It was introduced for the λ-calculus in [24]. Some important further developments of the notion can be traced in [21,16,5,18,27,23,30,2]. Owing to the tight connexion between permutation equivalence and concurrency, permutation equivalence appears in many guises in areas where concurrency is important. For instance, in trace theory the notion of Mazurkiewicz trace as a trace *up to the order of independent actions* was introduced in [29]. For another example, in transaction processing systems (TPSs: [12]) concurrency control protocols must prevent interference by ensuring so-called serialisability: a TPS should only accept a schedule if it is *equivalent* to some serial schedule. Roughly speaking, the notions of *conflict* and *view* equivalence in TPSs correspond to the notions of permutation and *causal* equivalence in term rewriting. Meseguer's rewriting logic ([36]) is a formalisation of this connexion in the case of first-order term rewriting. From the perspective of rewriting logic, the present paper can be seen as an investigation into its proof theory.

4 Parallel standardisation equivalence

The goal of *parallel standardisation* is to transform any reduction into a unique representative parallel standard reduction. Then, two reductions are said to be *parallel standardisation* equivalent if they yield the same parallel standard reduction. Since our transformation is effective, parallel standardisation equivalence is decidable.

The idea of the parallel standardisation procedure is to sort the steps in a reduction into outside-in order. We employ a parallel standardisation algorithm corresponding to inversion sort (see e.g. [48] Section 3.1).

Example 4.1 Consider the list $[3,2,1]$ of natural numbers and the natural ordering $<$. Inversion sort iteratively permutes adjacent pairs of members in the list which are in the wrong order, so-called *inversions*. Executed on the example list, we note there are two inversions: $[\underline{3,\overline{2}}\overline{,1}]$. Say the leftmost inversion, $\underline{3,2}$, is selected. After permutation the list contains one inversion $[2,\underline{3,1}]$. After permuting 3 and 1, the list contains yet another inversion $[\underline{2,1},3]$. Permuting it as well finally yields the sorted list $[1,2,3]$, not containing any inversions.

Similarly, parallel standardisation by inversion consists in repeatedly and non-deterministically permuting any pair of adjacent steps which are in the wrong, inside-out, order. Such pairs are called *anti-standard* pairs ([21]) and a reduction where no anti-standard pairs remain, is called *parallel standard*. The permutation process can be most easily described as a rewriting process, where the rules are oriented versions of the inner and outer permutation identities, which are applied modulo the structural equivalence induced by the remaining permutation identities (see Table 1 and Definition 3.2). The structural equivalence allows one to move annoying intermediate parallel reduction out of the way, in order to make steps which are in the wrong order adjacent.

Example 4.2 Reconsider the first reduction of our running example:

$$f(\tilde{a},a) \to f(b,\overline{a}) \to \underline{f}(b,\underline{b}) \to \underline{g}(\underline{b}) \to c$$

The first and third steps are in the wrong, inside-out, order, but not adjacent. Structural equivalence allows one to move the intermediate second step out of the way, by swapping the first and second step resulting in:

$$f(a,\overline{a}) \to f(\tilde{a},b) \to \underline{f}(b,\underline{b}) \to \underline{g}(\underline{b}) \to c$$

Now the second and third steps constitute an anti-standard pair: they are adjacent and in the wrong, inside-out order. Permuting them results in the second reduction of our running example:

$$f(a,\overline{a}) \to \underline{f}(a,\underline{b}) \to g(\tilde{a}) \to \underline{g}(\underline{b}) \to c$$

Note that this reduction is parallel standard, as it contains no anti-standard pairs (modulo structural equivalence). From this we conclude that the reductions are parallel standardisation equivalent.

We formalise our parallel standardisation procedure by inversion by means of a TRS, generated by the outer and inner identities, on proof terms modulo structural equivalence. As it turns out, parallel standard *proof terms* are parallel *reductions*, i.e. reductions consisting of parallel steps.

Example 4.3 Formalising Example 4.2 in this way yields:

$$\begin{aligned}\mathcal{R} &= \underline{f(\varrho, a) \cdot f(b, \varrho)} \cdot \vartheta(b) \cdot \varsigma \\ &\equiv f(a, \varrho) \cdot \underline{f(\varrho, b) \cdot \vartheta(b)} \cdot \varsigma \\ &\Rrightarrow f(a, \varrho) \cdot \vartheta(a) \cdot g(\varrho) \cdot \varsigma \\ &= \mathcal{S}\end{aligned}$$

where we first have swapped the first and second step using structural equivalence and then have applied a parallel standardisation step $\Rrightarrow$ (see Definition 4.7) to remove the anti-standard pair between the second and third step.

Since we perform our parallel standardisation steps *modulo* structural equivalence, we first show that structural equivalence is decidable, by computing canonical representatives of structural equivalence classes.

4.1 Canonisation

We will define canonical representatives of structural equivalence classes. This is achieved by first orienting the structural identities into rules of a TRS and then completing it into a terminating and confluent *canonisation* TRS. The idea of the orientation chosen here, ordering the functorial identity from left to right, is to 'push compositions inside' as far as possible. [5]

Definition 4.4 The rules of the *canonisation* TRS on proof terms are:

$$\begin{aligned}\circlearrowleft \cdot x &\Rightarrow x \\ x \cdot \circlearrowleft &\Rightarrow x \\ (x \cdot y) \cdot z &\Rightarrow x \cdot (y \cdot z) \\ f(x_1, \ldots, x_n) \cdot f(y_1, \ldots, y_n) &\Rightarrow f(x_1 \cdot y_1, \ldots, x_n \cdot y_n) \\ f(x_1, \ldots, x_n) \cdot (f(y_1, \ldots, y_n) \cdot z) &\Rightarrow f(x_1 \cdot y_1, \ldots, x_n \cdot y_n) \cdot z\end{aligned}$$

where f ranges over the (ordinary) function symbols. A proof term is *canonical*, if it is in canonisation normal form.

Example 4.5 The proof terms of Example 3.3 yield the same canonical proof term $g(\varrho, \varrho)$:

$$\begin{aligned}\overline{g(\varrho, a) \cdot g(b, \varrho)} &\Rightarrow g(\overline{\varrho \cdot b}, \overline{a \cdot \varrho}) \\ &\Rightarrow g(\varrho, \varrho) \\ &\Leftarrow g(\underline{a \cdot \varrho}, \underline{\varrho \cdot b}) \\ &\Leftarrow \underline{g(a, \varrho) \cdot g(\varrho, b)}\end{aligned}$$

[5] See [47] for a complete TRS obtained by orienting the functorial identity in the opposite direction.

The canonical proof terms for the reductions in our running example are distinct: $f(\varrho, \varrho) \cdot \vartheta(b) \cdot \varsigma$ and $\mathcal{S}$ itself.

Theorem 4.6 *Canonical proof terms are unique representatives of structural equivalence classes.*

Proof. It suffices to prove that the canonisation TRS is complete.

To see that canonisation is terminating, use the fact that the last two rules increase sharing, i.e. decrease the number of (ordinary) function symbols, and the observation that all rules are (left- and right-)linear.

To see that it is confluent, observe that the final rule (schema) is obtained by Knuth–Bendix completion of the first four (which directly correspond to the structural identities). □

It follows that computing unique representatives of structural equivalence classes is easy, hence that deciding structural equivalence is easy as well.

4.2 Parallel standardisation

Definition 4.7 The rules of the *parallel standardisation* TRS on proof terms are, for $\varrho : l \to r$, $x_i : s_i \geq t_i$

$$l(x_1, \ldots, x_n) \cdot \varrho(t_1, \ldots, t_n) \Rightarrow \varrho(x_1, \ldots, x_n)$$
$$\varrho(x_1, \ldots, x_n) \Rightarrow \varrho(s_1, \ldots, s_n) \cdot r(x_1, \ldots, x_n)$$

These rules are applied to proof terms modulo structural equivalence. [6]

Example 4.8 Consider $f(\underline{a}) \to \underline{f}(b) \to g(b)$ in the TRS with rules $\{\varrho : a \to b, \vartheta : f(x) \to g(x)\}$. It is witnessed by the proof term $\phi = f(\varrho) \cdot \vartheta(b)$. Clearly, ϕ is a redex for the first, *inner*, rule, and rewriting it yields $\psi = \vartheta(\varrho)$. Clearly, ψ is a redex for the second, *outer* rule, and rewriting it yields $\chi = \vartheta(a) \cdot g(\varrho)$.

Note that although χ in the example is parallel standard in the intuitive sense, it is not in parallel standardisation normal form since $\vartheta(a)$ is an outer redex. However, it fails only in a trivial way: $\chi \Rightarrow \vartheta(a) \cdot g(a) \cdot g(\varrho) \equiv \chi$. We simply forbid such trivial inner and outer steps.

Definition 4.9 A *parallel standardisation step* is a step in the parallel standardisation TRS, such that at least one of the variables substituted for in the applied rule, is not structurally equivalent to a loop. A proof term is *parallel standard* if it is in normal form w.r.t. parallel standardisation. Two proof terms are *parallel standardisation* equivalent if they have the same parallel standaridsation normal form modulo structural equivalence.

[6] Note that we do *not* assume that our proof terms are canonical, when applying a parallel standardisation rule.

Note that not being structurally equivalent to a loop is equivalent to containing some rule symbol. Therefore parallel standard reductions 'are' parallel reductions, i.e. sequences of parallel steps, whence their name.

Lemma 4.10 *Parallel standardisation is complete.*

Proof. The proof proceeds by showing termination and local confluence, both of which are complex.

The proof of local confluence proceeds by an analysis of the critical pairs between the inner and outer rules. The analysis is complicated by the fact that parallel standardisation steps are performed modulo structural equivalence. One proceeds by a case analysis on the relative positions of rule symbols in the canonical form of a proof term.

Randomly permuting inversions is inefficient already for lists, but permuting anti-standard pairs is more inefficient. This is caused by the fact that the term rewrite systems considered are non(-right)-linear, whereas sorting is linear. The idea of our termination proof is due to Klop ([21]) and is best explained via the following method for showing termination of inversion sort. Consider a list of say natural numbers on which sorting, say in increasing order, by permuting inversions would not terminate. Since there are only finitely many permutations of that list, non-termination implies that the sorting process cycles on some list. Consider a list, e.g. $[5, 3, \ldots]$, on the cycle. Then either

- the first element of the list is not involved in any permutation along the cycle, but then we have a shorter list ($[3, \ldots]$) on which sorting cycles by just omitting this first element, or
- the first element is involved in a permutation along the cycle, but then it is replaced by a smaller element ($[3, 5, \ldots]$). By well-foundedness of the less-than order, the first element will never be replaced by a bigger element by any permutation and a contradiction with cyclicity follows.

In the case of parallel standardisation, this argument is complicated by the fact that the TRSs considered are non-right-linear. Hence permutations are not length-preserving. Fortunately, the lengths of the proof terms obtainable by parallel standardisation can be bounded, by an appeal to the so-called *Finite Family Developments* theorem ([40]). □

The following theorem follows immediately from completeness of parallel standardisation, and the fact that parallel standardisation is defined by means of a TRS.

Theorem 4.11 (i) *Parallel standardisation equivalence is a congruence.*

(ii) *If ϕ and ψ are parallel standardisation equivalent, then* $\mathsf{src}(\phi) = \mathsf{src}(\psi)$ *and* $\mathsf{tgt}(\phi) = \mathsf{tgt}(\psi)$, *and similarly for reductions (i.e. modulo the reduction identities).*

Remark 4.12 (i) Structural equivalence is studied in an abstract axiomatic setting in [30] under the name of *square* permutation equivalence.

(ii) The results of the early days ([9,15,21]) mainly concern ordinary standardisation, i.e. sorting steps in textual left-to-right order, the reason being that both the λ-calculus, the prototypical higher-order term rewrite system, and combinatory logic, the prototypical first-order term rewrite system, are left-normal orthogonal systems. The methods presented above are also applicable to ordinary standardisation, as shown in [47]. The introduction of parallel standardisation is of more recent origin ([16]) and its further development was triggered by the axiomatic approaches to standardisation of [11,19,34].

(iii) Termination of parallel standardisation by inversion is essentially more difficult than termination of parallel standardisation by selection, corresponding to selection sort (see e.g. [48] Section 3.1 for selection sort, and [47] for 'selection (parallel) standardisation'). This is witnessed by the fact that the standardisation axioms in [30] do guarantee termination of the latter, while at the same time allowing for non-termination of the former.

5 Labelling equivalence

In *labelling* the behaviour of a reduction is seen from the point of view of an internal observer. Labelling is a transformation on rewrite systems which preserves their dynamics, but such that information about the reduction toward a target may be recorded in the target itself. Hence labelling a reduction $\mathcal{R}$ yields a (unique) reduction $\mathcal{R}'$ in which the same steps as in $\mathcal{R}$ are performed, but where the objects may contain extra information.

Example 5.1 Consider a very simple TRS for modelling a stack of natural numbers. Its signature consists of a nullary top-of-stack symbol $\top$ and unary stack-element symbols n, for $n \in \mathbb{N}$, and its rules are $\{\mathsf{push}^n : \top \to n(\top), \mathsf{pop}^n : n(\top) \to \top \mid n \in \mathbb{N}\}$. An example of a reduction starting from the empty stack $\top$ in this TRS is

$$\mathcal{R} : \top \to 5(\top) \to 5(1(\top)) \to 5(\top) \to 5(3(\top))$$

Suppose now we want to record the information of how many numbers there are on the stack, in the top-of-stack symbol $\top$. To that end we can label the symbol by a natural number i, say as $\top_i$, and change the TRS into $\{\mathsf{push}^n_i : \top_i \to n(\top_{i+1}), \mathsf{pop}^n_i : n(\top_{i+1}) \to \top_i \mid n, i \in \mathbb{N}\}$. After we have chosen a way to label the source $\top$ of $\mathcal{R}$, say as $\top_0$, labelling $\mathcal{R}$ yields a unique labelled reduction:

$$\mathcal{R}' : \top_0 \to 5(\top_1) \to 5(1(\top_2)) \to 5(\top_1) \to 5(3(\top_2))$$

Two reductions are defined to be *labelling* equivalent if their (labelled) targets are the same for any labelling of their sources. An important instance of labelling is the so-called *Lévy* labelling $\mathfrak{L}$, which records the *complete* history of *each* symbol along a reduction into the label of that symbol itself. This implies that every other labelling can be obtained from the Lévy labelling by 'forgetting' part of the history recorded. Hence in order to study labelling equivalence it suffices to study Lévy labelling equivalence.

Example 5.2 To keep the Lévy labellings $\mathfrak{L}(\mathcal{R})$ and $\mathfrak{L}(\mathcal{S})$ (see Definition 5.20) of the reductions $\mathcal{R}$ and $\mathcal{S}$ of our running example readable and legible, we have removed as much redundant information as possible, yielding

$$\mathfrak{L}(\mathcal{R}) : f(\tilde{a}, a) \to f(b^{\tilde{a}}, a) \to f(b^{\tilde{a}}, b^{a}) \to g^{f(x,b^a)}(b^{\tilde{a}}) \to c^{g^{f(x,b^a)}(b^{\tilde{a}})}$$
$$\mathfrak{L}(\mathcal{S}) : f(\tilde{a}, a) \to f(\tilde{a}, b^{a}) \to g^{f(x,b^a)}(\tilde{a}) \to g^{f(x,b^a)}(b^{\tilde{a}}) \to c^{g^{f(x,b^a)}(b^{\tilde{a}})}$$

Note that the target c of $\mathcal{R}$ and $\mathcal{S}$ has received the same Lévy label $g^{f(x,b^a)}(b^{\tilde{a}})$, hence the original reductions are Lévy labelling equivalent and therefore labelling equivalent.

5.1 Abstract rewrite labelling

We define labelling as adjoining or affixing information to a rewrite system. Of course, this should not interfere with the dynamics of the system, i.e. with rewriting. That is, the systems before and after the labelling should have the same behaviour. We formalise the behavioural equivalence via a notion of bisimulation.

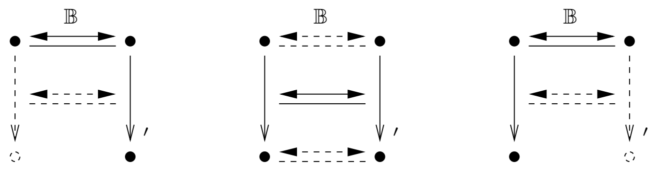

Fig. 5. Bisimulation: back, relator, forth

Definition 5.3 A relation $\mathbb{B}$ is a *bisimulation* relation between abstract rewrite systems $\to$ and $\to'$ if objects are related to objects and steps are related to steps, such that

- if $a \mathbin{\mathbb{B}} a'$, then $\mathbb{B}$ relates each step from a' to some step from a (*back*) and each step from a to some step from a' (*forth*),
- if $\phi \mathbin{\mathbb{B}} \phi'$ with $\phi : a \to b$ and $\phi' : a' \to' b'$, then $a \mathbin{\mathbb{B}} a'$ and $b \mathbin{\mathbb{B}} b'$ (*relator*).

Related objects or steps are called $\mathbb{B}$-*bisimilar* (see Figure 5, where $\underline{\leftrightarrow}$ denotes bisimilarity).

Objects (steps) are *bisimilar* if they are bisimilar for some bisimulation.

Example 5.4 A bisimulation $\mathbb{B}$ between the black hole ARS $\circlearrowright$ and (an isomorphic copy $\to'$ of) the infinite straight line ARS $\multimap\!\infty\!\to$ (see Example 2.3) is given by defining, for all $i \in \mathbb{N}$ (see Figure 6, where dotted lines indicate $\mathbb{B}$-related objects and steps)

- on objects $\bullet \mathbin{\mathbb{B}} \bullet_i$, and

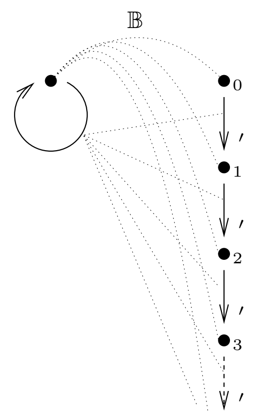

Fig. 6. Example of a bisimulation

– on steps $\mathbb{B}$ relates $\bullet \to \bullet$ to $\bullet_i \to' \bullet_{i+1}$.

Labellings are a special, one-way case of bisimulations. Whereas bisimulation is symmetric in relating two abstract rewrite systems, labelling is asymmetric. The idea is that although there can be more than one way to label an object or a step, distinct objects and steps cannot be identified with each other by labelling them. The second part of this idea can be stated contrapositively as: to every labelled object or step corresponds a unique unlabelled object or step.

Definition 5.5 Let $\mathbb{B}$ be a bisimulation between $\to$ and $\to'$.

(i) $\mathbb{B}$ is a *labelling* (*of* $\to$ *to* $\to'$), if

(1) for every a' there is a unique a such that $a \mathbin{\mathbb{B}} a'$, and

(2) to every $\phi : a \to b$ and $a \mathbin{\mathbb{B}} a'$, corresponds a unique $\phi' : a' \to' b'$ such that $\phi \mathbin{\mathbb{B}} \phi'$. The correspondence must be bijective and the step is denoted by $\mathbb{B}_{a'}(\phi)$ and called the *a'-label* of ϕ.

(ii) A *rewrite* labelling $\mathfrak{B}$ is a pair consisting of a labelling $\mathbb{B}$ together with an *initial* labelling function $\mathfrak{b}$ mapping objects of $\to$ to bisimilar objects of $\to'$. That is, $a \mathbin{\mathbb{B}} \mathfrak{b}(a)$ for all objects a. We denote the label $\mathbb{B}_{\mathfrak{b}(a)}(\phi)$ of the step $\phi : a \to b$ by $\mathfrak{B}(\phi)$.

This is visualised in Figure 7, where $\underline{\leftarrow}$ denotes labelling and ! denotes uniqueness.

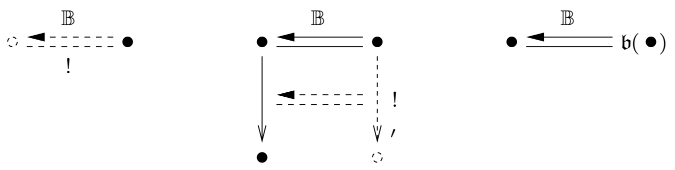

Fig. 7. Rewrite labelling: inversely functional, unique labelling, initial label

Note that the inverse of a labelling $\mathbb{B}$ is a function on objects by uniqueness condition (1). By uniqueness condition (2) related objects look the same *qua* outgoing steps.

Example 5.6 In the reduction $\mathcal{R}$ of Example 5.1 we have chosen to label its source (the empty stack) by 0. This fixed the labelling of all steps in $\mathcal{R}'$.

Choosing the smallest types (according to some well-order on types) for Church-typing a Curry-typable term makes Church-typing into a rewrite labelling. Note that for a Curry-typable λ-term, say $(\lambda x.x)y$, there are still many ways to enrich it with Church-typing, e.g. $(\lambda x{:}o.x)y$ or $(\lambda x{:}o{\to}o.x)y$. However, choosing a specific one, e.g. the smallest one, fixes how the terms in any reduction from the given term are to be typed.

Since bisimulation and hence labelling are behaviour preserving, one expects that rewrite properties can be transferred along them. This is indeed the case.

Proposition 5.7 *If $a \mathbb{B} b$ for some bisimulation $\mathbb{B}$, then*

- *a is normalizing (*WN*) iff b is normalizing,*
- *a is terminating (*SN*) iff b is terminating, and*
- *if $\mathbb{B}$ is a labelling, then the diamond property of b implies the diamond property of a (see Figure 8), but not the other way around.*

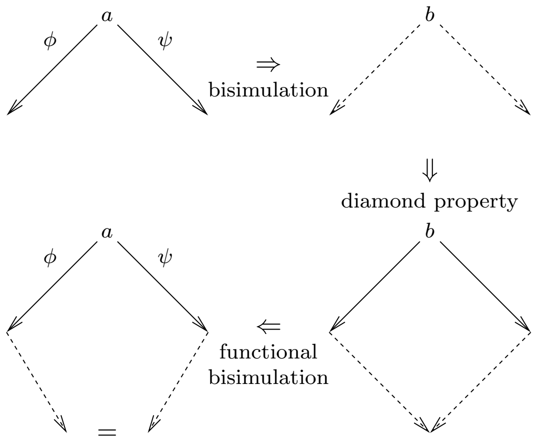

Fig. 8. Diamond property of a via the diamond property of b

Definition 5.8 Let $\mathfrak{B}$ be a rewrite labelling. Two co-initial reductions[7] are said to be $\mathfrak{B}$-*labelling* equivalent if their $\mathfrak{B}$-labelled targets are the same. If this holds for any term rewrite labelling $\mathfrak{A}$, then the reductions are said to be *labelling* equivalent.

[7] We have not defined reductions for ARSs. However, reductions could be defined as TRS reductions by viewing the objects as (nullary) symbols and the steps as rules of a TRS. Labelling reductions is then defined via labelling of proof terms, as defined below.

We now show that labelling equivalence can be checked by checking labelling equivalence for a particular, maximal, labelling called *unwinding*. The unwinding is maximal w.r.t. the so-called *history* order. Two rewrite labellings of the same abstract rewrite system can be compared in the obvious way: if the former can be post-composed with a third labelling to yield the latter, the former is less than or equal in the history order to the latter.

Definition 5.9 Let $\to$ be an abstract rewrite system. We define the *history* order $\leq$ on labellings $\mathfrak{A}$[8] and $\mathfrak{B}$ of $\to$ by: $\mathfrak{A} \leq \mathfrak{B}$ if there is some labelling $\mathfrak{C}$, such that $\mathfrak{B}$ is the composition of $\mathfrak{A}$ and $\mathfrak{C}$, for objects reachable from initial labels. A labelling is *maximal* if it is maximal with respect to the history order.

Example 5.10 Consider the abstract rewrite system having three objects a, b, and c and four steps $\phi_i : a \to b$ and $\psi_i : b \to c$ for $i \in \{1, 2\}$ and consider the following three labellings:

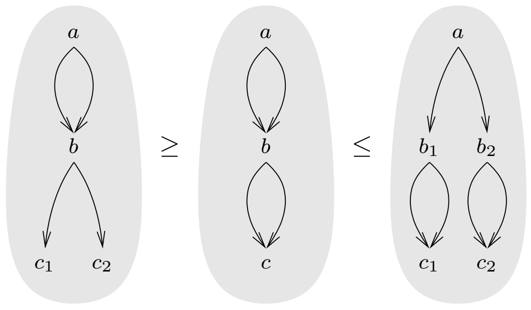

Fig. 9. The history order on labellings

- The identity labelling as shown in the middle of Figure 9.
- The labelling which distinguishes between ψ_1 and ψ_2 as shown on the left of Figure 9.
- The labelling which distinguishes between ϕ_1 and ϕ_2 as shown on the right of Figure 9.

The second and third labellings are incomparable in the history order, but both are better than the identity labelling.

For any abstract rewrite system, there is a maximal labelling, which we call unwinding. The intuitive reason why a maximal labelling exists is that labels can be used to ‘split’ the targets of a pair of cofinal steps. Once all steps have distinct targets nothing more can be done.

Example 5.11 Applying this process to the ARS of Example 5.10 yields the ARS displayed in Figure 10, which is easily seen to be maximal with respect to the history order $\leq$.

[8] Beware, the symbols $\mathfrak{A}$ and $\mathfrak{U}$ (fraktur A and U) are not to be confused.

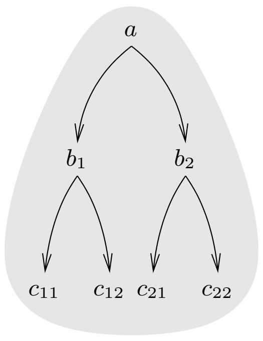

Fig. 10. Unwinding of the ARS of Example 5.10

The trivial idea of the unwinding $\mathfrak{U}$ is to achieve this splitting by adjoining the history of objects to the objects themselves (which obviously causes the targets of cofinal steps to be distinct).

Definition 5.12 Let $\rightarrow = \langle A, \Phi \rangle$ be an ARS. We simultaneously construct an ARS $\mathfrak{U}(\rightarrow) = \langle \mathfrak{U}(A), \mathfrak{U}(\Phi) \rangle$ and a rewrite labelling $\mathfrak{U} = \langle \mathbb{U}, \mathfrak{u} \rangle$ such that $\mathfrak{U}(\rightarrow)$ is the $\mathfrak{U}$-labelling of $\rightarrow$.

Any object a in A is an object of $\mathfrak{U}(A)$ as well. It is bisimilar to itself ($a \mathbb{U} a$) and its own initial label ($\mathfrak{u}(a) = a$).

Suppose ϕ is a step in Φ with $\phi : a \rightarrow b$, and $a \mathbb{U} a'$. Then we let $\phi_{a'}$ be both an object and a step of $\mathfrak{U}(\rightarrow)$ ($\phi_{a'} \in \mathfrak{U}(A), \mathfrak{U}(\Phi)$) such that the step ends in the object: $\phi_{a'} : a' \rightarrow \phi_{a'}$. As a step $\phi_{a'}$ is a label of ϕ ($\phi \mathbb{U} \phi_{a'}$) and as an object it is a label of the target b of ϕ ($b \mathbb{U} \phi_{a'}$).

The abstract rewrite system $\mathfrak{U}(\rightarrow)$ is called the *unwinding* of $\rightarrow$.

Note that $\phi_{a'}$ serves both as a step and as an object in the unwinding. This is easily explained by noting that it can be thought of as the unique access path to both the step and its target.

Lemma 5.13 *Unwinding is maximal with respect to the history order.*

Theorem 5.14 *Two co-initial reductions are labelling equivalent iff the targets of their unwindings are the same.*

Since the unwinding of an ARS has 'perfect recall', we have as an easy consequence that two reductions are labelling equivalent iff they are identical. For TRSs the situation will become more interesting.

5.2 Term rewrite labelling

Definition 5.15 A *term rewrite* labelling $\mathfrak{B}$ is a labelling between term rewrite systems, consisting of a labelling $\mathbb{B}$ on its signatures and rules and an *initial* labelling $\mathfrak{b}$ mapping terms to labelled terms.

A *signature* labelling is an injection between signatures respecting arities. We use α, β, ... to range over labels of function symbols. The signature labelling naturally induces a labelling of terms over the signatures.

A *rewrite rule* labelling is a labelling between (abstract rewrite systems representing) rules, respecting the signature labelling and such that variables are related (only) to themselves.

Note that the injection on signatures can be construed as a labelling in the technical sense, by viewing both signatures as single-object abstract rewrite systems having their symbols as loops.

Example 5.16 (i) The simplest example of a term rewrite labelling is the identity labelling, relating symbols, terms, and rules to themselves.

(ii) Any choice of initial labelling will make the labelling of Example 5.1 into a term rewrite labelling.

Note that the composition of two term rewrite labellings is a term rewrite labelling again, simply by composing componentwise (i.e. the signature, the rule labellings and the initial labellings). Moreover, any term rewrite labelling between term rewrite systems induces a canonical *underlying* rewrite labelling between the respective underlying abstract rewrite systems.

$$\begin{tikzcd}
\text{term rewriting system} \arrow[d, "\text{underlying}"] & \mathcal{T} \arrow[d] & \mathcal{U} \arrow[l, Rightarrow, "\mathfrak{A}"'] \arrow[d] \\
\text{abstract rewriting system} & \geq_{\mathcal{T}} & \geq_{\mathcal{U}} \arrow[l, dashed, "\geq_{\mathfrak{A}}"']
\end{tikzcd}$$

Fig. 11. Term rewrite labelling induces abstract rewrite labelling

Definition 5.17 Let $\mathfrak{A}$ be a term rewrite labelling between term rewrite systems $\mathcal{T}$ and $\mathcal{U}$, consisting of a labelling $\mathbb{A}$ and an initial labelling $\mathfrak{a}$. The *underlying* rewrite labelling $\geq_{\mathfrak{A}}$ between the underlying abstract rewrite systems $\geq_{\mathcal{T}}$ and $\geq_{\mathcal{U}}$ (see Figure 11) is defined as follows.

The label $\mathfrak{A}(\phi)$ of a proof term ϕ is the unique ϕ' such that $\phi \; \mathbb{A} \; \phi'$ and $\mathfrak{a}(\mathsf{src}(\phi)) = \mathsf{src}(\phi')$, where $\mathbb{A}$ on proof terms is defined by:

– $\phi \cdot \psi \; \mathbb{A} \; \phi' \cdot \psi'$, if $\phi \; \mathbb{A} \; \phi'$ and $\psi \; \mathbb{A} \; \psi'$.

– $f(\phi_1, \ldots, \phi_n) \; \mathbb{A} \; f'(\phi'_1, \ldots, \phi'_n)$, if $f \; \mathbb{A} \; f'$ and $\phi_i \; \mathbb{A} \; \phi'_i$, for all $1 \leq i \leq n$.

– $\varrho(\phi_1, \ldots, \phi_n) \; \mathbb{A} \; \varrho'(\phi'_1, \ldots, \phi'_n)$, if $\varrho \; \mathbb{A} \; \varrho'$ and $\phi_i \; \mathbb{A} \; \phi'_i$, for all $1 \leq i \leq n$.

Unique existence is easily shown by induction on proof terms.

Definition 5.18 Let $\mathfrak{A}$ be a term rewrite labelling. Two co-initial proof terms are said to be $\mathfrak{A}$-*labelling* equivalent if their $\mathfrak{A}$-labelled targets are the same. If this holds for any term rewrite labelling $\mathfrak{A}$, then the proof terms are said to be *labelling* equivalent.

Theorem 5.19 (i) *Labelling equivalence is a congruence.*

(ii) *If ϕ and ψ are labelling equivalent, then* $\mathsf{src}(\phi) = \mathsf{src}(\psi)$ *and* $\mathsf{tgt}(\phi) = \mathsf{tgt}(\psi)$*, and similarly for reductions (i.e. modulo the reduction identities).*

Proof. That labelling equivalence is an equivalence relation follows by 'sameness' being an equivalence relation. That it is a congruence follows from the inductive definition of labelling of proof terms. The second item is trivial. □

The history order on term rewrite labelling is defined as for abstract rewrite labelling above, but restricted to term rewrite labellings. We now introduce Lévy labelling. It plays a rôle similar to that of unwinding for ARSs: it maintains complete information about the reduction history of terms. This causes it to be a maximal term rewrite labelling w.r.t. the history order, which in turn allows us to reduce labelling equivalence to Lévy labelling equivalence. Technically, maintaining complete information is achieved by 'putting' the history of a left-hand side, i.e. its collection of labels, on all function symbols in the right-hand side. Since these symbols need to be distinguishable from one another, each such symbol will moreover be labelled by its unique position in the right-hand side.

Definition 5.20 Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a term rewrite system. The symbols and rules of the *Lévy* labelling $\mathfrak{L}(\mathcal{T}) = \langle \mathfrak{L}(\Sigma), \mathfrak{L}(R) \rangle$of $\mathcal{T}$ and the bisimulation $\mathbb{L}$ between $\mathcal{T}$ and $\mathfrak{L}(\mathcal{T})$ are defined by simultaneous induction (see Figure 12):

– Suppose p is a position in the term s. Then $s^p \in \mathfrak{L}(\Sigma)$ and $s(p) \mathbb{L} s^p$.
– Suppose $\varrho \in R$ with $\varrho : l \to r$, and $l \mathbb{L} l'$. Then for every non-variable position p in r, $\varrho^p_{l'} \in \mathfrak{L}(\Sigma)$ and $r(p) \mathbb{L} \varrho^p_{l'}$. Furthermore, $\varrho_{l'} \in \mathfrak{L}(R)$ and $\varrho \mathbb{L} \varrho_{l'}$ with $\varrho_{l'} : l' \to r'$, where r' is the Lévy labelling of r such that $r'(p) = \varrho^p_{l'}$, for every non-variable position p in r.

The initial labelling $\mathfrak{l}$ labels all symbols by their position: $\mathfrak{l}(s) = \mathfrak{l}_\varepsilon(s)$, where for any position p, $\mathfrak{l}_p(f(s_1, \ldots, s_n)) = s^p(\mathfrak{l}_{1\cdot p}(s_1), \ldots \mathfrak{l}_{n\cdot p}(s_n))$.

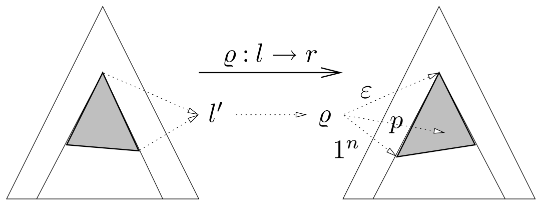

Fig. 12. Lévy labelling

Example 5.21 Consider the TRS $\{\varrho : a \to a\}$ and the reduction $\mathcal{R} : a \to a \to a \to a \to \cdots$. The Lévy labelling of the TRS has rules such as $\varrho_{a^\varepsilon} : a^\varepsilon \to \varrho^\varepsilon_{a^\varepsilon}$ and $\varrho_{f(a)^1} : f(a)^1 \to \varrho^\varepsilon_{f(a)^1}$ for rewriting atomic labels, and rules such as $\varrho_{\varrho^\varepsilon_{a^\varepsilon}} : \varrho^\varepsilon_{a^\varepsilon} \to \varrho^\varepsilon_{\varrho^\varepsilon_{a^\varepsilon}}$ for rewriting complex labels. Lévy labelling $\mathcal{R}$ yields the labelled reduction $\mathfrak{L}(\mathcal{R}) : a^\varepsilon \to \varrho^\varepsilon_{a^\varepsilon} \to \varrho^\varepsilon_{\varrho^\varepsilon_{a^\varepsilon}} \to \varrho^\varepsilon_{\varrho^\varepsilon_{\varrho^\varepsilon_{a^\varepsilon}}} \to \cdots$.

Lemma 5.22 *Lévy labelling is maximal with respect to the history order.*

Theorem 5.23 *Labelling equivalence coincides with Lévy labelling equivalence.*

Remark 5.24 (i) Labelling in rewriting goes back at least to Newman ([38]) and is employed in one way or another, e.g. in the form of underlining or colouring or as semantic labelling, in many papers throughout the literature. We are however unaware of a general approach to labelling for rewriting based on bisimulation, as put forward here. (But cf. [26] for a notion of labelled term rewrite system having the same purpose as the present one, namely characterizing permutation equivalence.)

(ii) The notion of bisimulation is originally due to Park ([42]) and to Milner ([37]). Labellings are closely related to functional bisimulations and bounded morphisms (also known as p-morphisms) as studied in e.g. [1,4]. However, note that steps in our abstract rewrite systems do not carry labels, and that we allow for several distinct steps between the same two objects.

(iii) Lévy labelling does *not* record the complete reduction history in case collapsing rules are applied. This is particularly clear for the term $c(a)$ in the collapsing TRS $\{c(x) \to c(x), c(x) \to x\}$: no matter how many steps according to the first rule are performed before applying the second rule, the Lévy label of the target will always be the same. We refer the reader to [47] for a solution to this problem.

6 Projection equivalence

A common way to do shopping is to have a shopping list and cross out each item on the list you put in your shopping cart, until there are no remaining/residual items on the list. A similar process can be carried out to establish the equality of the lists $[2,1,0]$ and $[0,1,2]$. To establish that the second list is contained in the first one, we cross out the successive elements of the first list from the second one:

$$[0,1,2] \rightsquigarrow_2 [0,1,\cancel{2}] \rightsquigarrow_1 [0,\cancel{1},\cancel{2}] \rightsquigarrow_0 [\cancel{0},\cancel{1},\cancel{2}]$$

Since no item remains, the second list is indeed contained in the first one. Similarly, the reverse holds:

$$[2,1,0] \rightsquigarrow_0 [2,1,\cancel{0}] \rightsquigarrow_1 [2,\cancel{1},\cancel{0}] \rightsquigarrow_2 [\cancel{2},\cancel{1},\cancel{0}]$$

hence we conclude that the lists are equivalent. Viewing the items of the list as steps, this process can be seen as a particular confluence procedure, as visualised in Figure 13. Distinct items commute (see e.g. the top left rectangle) since crossing out an item from a list headed by a different item means crossing it out from the tail of the list. Identical items (see e.g. the top right rectangle) cancel each other out, yielding loop steps. Since the right-hand side of the diagram consists of loop steps, we say that the list $[0,1,2]$ is less than or equal to $[2,1,0]$ in the *projection* order. In fact, the two lists are *projection*

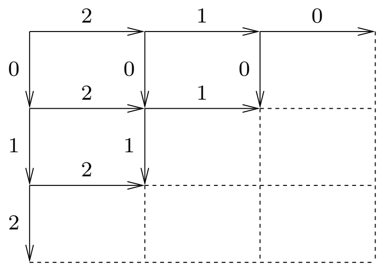

Fig. 13. Projection equivalence of $[2,1,0]$ and $[0,1,2]$

equivalent, since both the right-hand side and the bottom of the diagram consists of loop steps. In this section, we show that for any left-linear TRS a notion of projection can be defined in a natural way. The idea is that the projection of a reduction over another, co-initial reduction yields a reduction consisting of steps which are performed in the former, but not in the latter reduction.

$$\begin{tikzcd}
 & & & \mathcal{S} & & & \\
 & f(a,a) \arrow[r] \arrow[d] & f(a,b) \arrow[r] \arrow[d, "f(\varrho,b)"] & g(a) \arrow[r] \arrow[d] & g(b) \arrow[r] \arrow[d, dash, dotted] & c \arrow[d, dash, dotted, "\circlearrowright"] & \\
 & f(b,a) \arrow[r] \arrow[d] & f(b,b) \arrow[r] \arrow[d, dash, dotted, "\circlearrowright"] & g(b) \arrow[r, dash, dotted] \arrow[d, dash, dotted] & g(b) \arrow[r] \arrow[d, dash, dotted] & c \arrow[d, dash, dotted, "\circlearrowright"] & \\
\mathcal{R} & f(b,b) \arrow[r, dash, dotted] \arrow[d] & f(b,b) \arrow[r] \arrow[d] & g(b) \arrow[r, dash, dotted] \arrow[d, dash, dotted] & g(b) \arrow[r] \arrow[d, dash, dotted] & c \arrow[d, dash, dotted, "\circlearrowright"] & \mathcal{R}/\mathcal{S} \\
 & g(b) \arrow[r, dash, dotted] \arrow[d] & g(b) \arrow[r, dash, dotted] \arrow[d] & g(b) \arrow[r, dash, dotted] \arrow[d] & g(b) \arrow[r] \arrow[d] & c \arrow[d, dash, dotted, "\circlearrowright"] & \\
 & c \arrow[r, dash, dotted, "\circlearrowright"'] & c \arrow[r, dash, dotted, "\circlearrowright"'] & c \arrow[r, dash, dotted, "\circlearrowright"'] & c \arrow[r, dash, dotted, "\circlearrowright"'] & c & \\
 & & & \mathcal{S}/\mathcal{R} & & &
\end{tikzcd}$$

Fig. 14. Residual of $\mathcal{R}$ after $\mathcal{S}$ and vice versa

Example 6.1 Computing the residual of $\mathcal{R}$ after $\mathcal{S}$ and vice versa will proceed as in Figure 14, but in a completely inductive way. Since this computation is easy but tedious, we will compute as an example $\mathcal{R}/\mathcal{S}$ only up to the second step of $\mathcal{S}$. In the computation we use the notation $\mathcal{P}_i$ to indicate the tail of a reduction $\mathcal{P}$ from the ith step on, so $\mathcal{P}_0 = \mathcal{P}$.[9]

$$\begin{aligned}
\mathcal{R}/\underline{\mathcal{S}} &= \underline{\mathcal{R}/(f(a,\varrho)\cdot\mathcal{S}_1)}\\
&= (\underline{\mathcal{R}}/f(a,\varrho))/\mathcal{S}_1\\
&= \underline{(f(\varrho,a)\cdot\mathcal{R}_1)/f(a,\varrho)}/\mathcal{S}_1\\
&= (\underline{f(\varrho,a)/f(a,\varrho)}\cdot(\mathcal{R}_1/\underline{f(a,\varrho)/f(\varrho,a)}))/\mathcal{S}_1\\
&= (f(\underline{\varrho/a},\underline{a/\varrho})\cdot(\mathcal{R}_1/f(\underline{a/\varrho},\underline{\varrho/a})))/\mathcal{S}_1\\
&= (f(\varrho,b)\cdot(\underline{\mathcal{R}_1}/f(b,\varrho)))/\mathcal{S}_1\\
&= (f(\varrho,b)\cdot\underline{(f(b,\varrho)\cdot\mathcal{R}_2)/f(b,\varrho)})/\mathcal{S}_1\\
&= (f(\varrho,b)\cdot\underline{f(b,\varrho)/f(b,\varrho)}\cdot(\mathcal{R}_2/\underline{f(b,\varrho)/f(b,\varrho)}))/\mathcal{S}_1\\
&= (f(\varrho,b)\cdot\circlearrowright\cdot\underline{\mathcal{R}_2/\circlearrowright})/\mathcal{S}_1\\
&= (f(\varrho,b)\cdot\circlearrowright\cdot\mathcal{R}_2)/\mathcal{S}_1
\end{aligned}$$

Note that the reduction $f(\varrho,b)\cdot\circlearrowright\cdot\mathcal{R}_2$ in the final expression corresponds exactly to the second column in Figure 14. Continuing the computation in this fashion eventually yields $\mathcal{R}/\mathcal{S} = \circlearrowright$, as desired. This shows that $\mathcal{R}$ is less than or equal to $\mathcal{S}$ in the projection order. We leave it to the reader to verify that $\mathcal{S}/\mathcal{R} = \circlearrowright$ holds as well, showing that, in fact, $\mathcal{R}$ and $\mathcal{S}$ are projection equivalent.

Remark 6.2 Although there is a tight correspondence between multi-steps and complete developments, the former are more intensional. In particular, as soon as one selects some order for developing the redexes in a multi-step, the cube identity is bound to fail. This can be seen as follows. Consider the three possible steps from the term $f(f(a))$ in the TRS with rules $\varrho : a \to b$ and $\vartheta : f(x) \to g(x,x)$. Computing the tilings obtained by contracting both f-redexes in either order gives rise to the 'scarab' in Figure 15. Now note that one way one ends up contracting the first and third or the second and fourth a-redexes first, and the other way the first and second or the third and fourth redexes, from which one concludes that the cube identity is violated (cf. [3] Exercise 12.4.5). Hence, we prefer working with multi-steps instead of ordinary steps, developing the former only at the latest possible moment into reductions consisting of ordinary steps.

6.1 *Abstract residual systems*

Definition 6.3 *A(n abstract) residual system* is a triple $\langle\to,\circlearrowright,/\rangle$, where $\to$ is an abstract rewrite system (Definition 2.2) and $/$ is a *residuation* for $\to$

[9] The rules employed here to compute residuals can be found in Definition 6.9(iii).

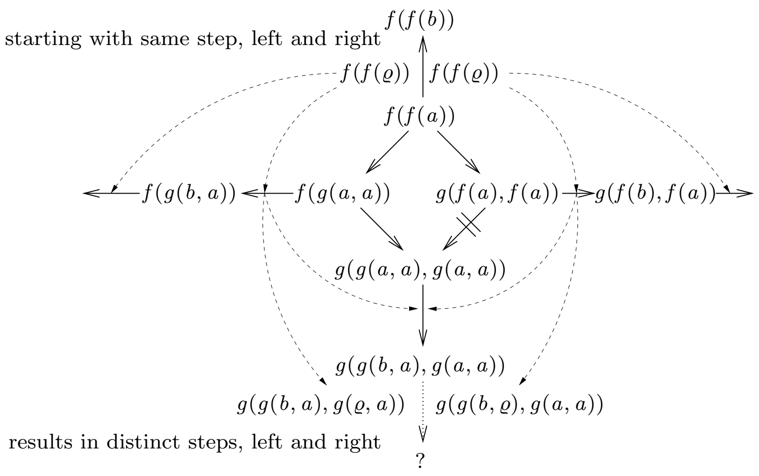

Fig. 15. Failure of cube identity for developments

having $\circlearrowright$ for *loop*. That is,

- $\circlearrowright$ is a function from objects to $\rightarrow$-loops, i.e. $\mathsf{tgt}(\circlearrowright_a) = a = \mathsf{src}(\circlearrowright_a)$,
- $/$ is a function from pairs of co-initial steps to steps such that $\mathsf{tgt}(\phi) = \mathsf{src}(\psi/\phi)$ and $\mathsf{tgt}(\phi/\psi) = \mathsf{tgt}(\psi/\phi)$ (Figure 17 top left).

It must satisfy the *residual* identities in Table 2.

$$\boxed{\begin{array}{c}(\phi/\psi)/(\chi/\psi) \approx (\phi/\chi)/(\psi/\chi)\\ \phi/\phi \approx \circlearrowright\\ \phi/\circlearrowright \approx \phi\\ \circlearrowright/\phi \approx \circlearrowright\end{array}}$$

Table 2
The cube and the loop identities of residual systems

We often identify a loop with its source (and target) object. A residuation will also be called a *residual* or *projection* operation. The first identity of Table 2 is called the *cube* identity and the other identities the *loop* identities (see Figure 16 for a two-dimensional rendering of the cube identity). Notions for abstract rewrite systems extend to residual systems via their underlying abstract rewrite systems.

Remark 6.4 The residual identities are not independent; the third loop identity is derivable from the others: $\circlearrowright/\phi = (\circlearrowright/\circlearrowright)/(\phi/\circlearrowright) = (\circlearrowright/\phi)/(\circlearrowright/\phi) = \circlearrowright$. This style of derivation is prototypical for derivations of identities in residual systems.

Several basic systems occurring in mathematics and computer science can

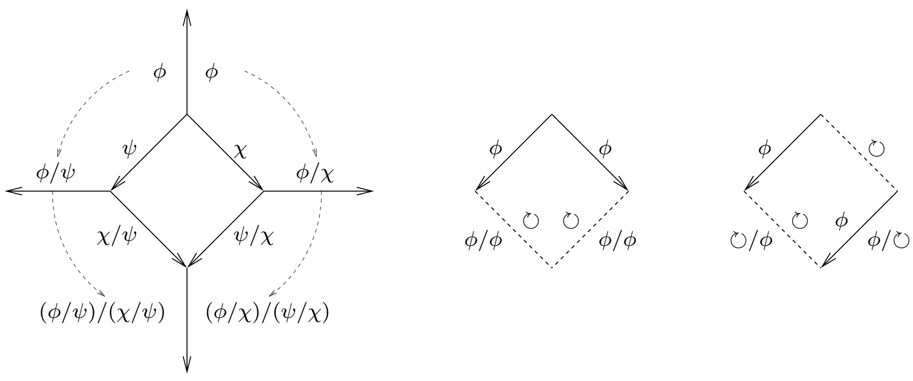

Fig. 16. The residual identities

be viewed as residual systems. In each case, the underlying abstract rewrite system consists of a single object, hence its steps are loops.

Example 6.5 (1) Consider the ARS having the natural numbers as loops. On this ARS the cutoff-subtraction $\ominus$ is a residual operation having 0 as loop. The only identity which is slightly non-trivial to check is the cube identity. It holds since

$$(n \ominus m) \ominus (k \ominus m) = n \ominus \max(m, k) = (n \ominus k) \ominus (m \ominus k)$$

Note that the natural numbers do not have a unary minus. (The integers with zero and subtraction do not constitute a residual system, since the third loop identity $0 - x = 0$ is clearly violated!)

(2) Consider the ARS having sets as loops. On this ARS set difference $-$ is a residual operation having the empty set $\emptyset$ as loop. The cube identity holds:

$$(A - B) - (C - B) = A - (B \cup C) = (A - C) - (B - C)$$

In words: first removing from A the elements of B and then removing the elements of C which had not been removed yet is the same as removing from A all elements which are in either B or C, i.e. the union of B and C.

(3) Consider the ARS having multisets as loops. On this ARS multiset difference $-_\#$ is a residual operation having the empty multiset $\emptyset_\#$ as loop. The cube identity holds as in the previous item, replacing set by multiset union $\cup_\#$.

Note that single-object residual systems are algebras, hence are closed under products.

Definition 6.6 Let $\mathbf{R} = \langle \rightarrow, \circlearrowright, / \rangle$ be a residual system. The *projection* order $\lesssim$ and the corresponding *projection* equivalence $\simeq$ are defined by

$$\begin{aligned} \phi \lesssim \psi &\quad \text{if } \phi/\psi = \circlearrowright \\ \phi \simeq \psi &\quad \text{if } \phi \lesssim \psi \text{ and } \psi \lesssim \phi \end{aligned}$$

for co-initial steps ϕ and ψ.

Notice that $\circlearrowleft \lesssim \phi$ always holds. We use $\phi < \psi$ to abbreviate '$\phi \lesssim \psi$, but not $\psi \lesssim \phi$'. Notions for orders extend to residual systems via their projection order, but note that only co-initial steps are ordered.

Example 6.7 (1) The projection order of the natural numbers is the usual less-than-or-equal partial order and projection equivalence is just the identity relation.

(2) The projection order on sets is set inclusion.

(3) The projection order on multisets is multiset inclusion.

Theorem 6.8 (1) *The projection order is a quasi-order.*

(2) *Projection equivalence is an equivalence relation, which is a congruence for the operations.*

6.2 Term residual systems

We show that proof terms can be made directly into a residual system in a natural way, such that permutation equivalence coincides with the projection equivalence of this residual system. The set-up is in the spirit of the inductive definition of proof terms. The ideas embodied by the identities for projecting composites are visualised in Figure 17. In order to compute residuals in case

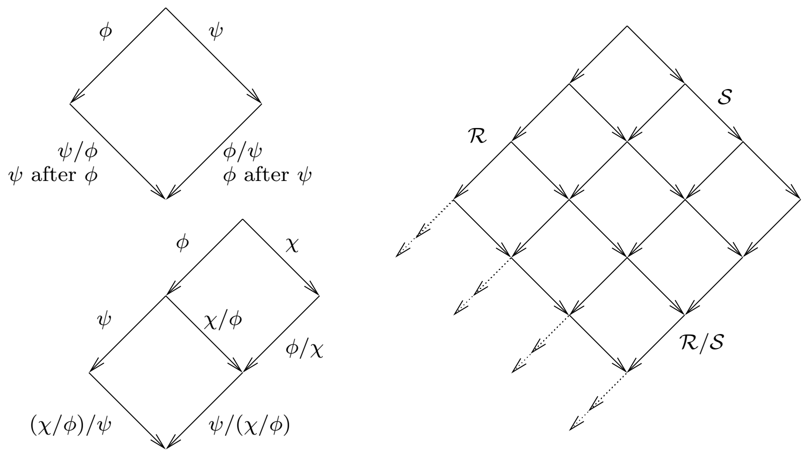

Fig. 17. Residuals of steps, composites and reductions

of conflict, we first introduce an error symbol and an error rule.

Definition 6.9 Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a left-linear TRS. The term rewrite system with *error* $\mathcal{T}_{\#} = \langle \Sigma_{\#}, R_{\#} \rangle$ is defined by adjoining a nullary *error* symbol # to Σ, and adjoining the *error* rule $\# : x \to \#$ to R.

(i) The *proof term* residual system $\hat{\mathcal{T}}$ has as abstract rewrite system $\geq_{\mathcal{T}_{\#}}$ modulo the loop and functorial identities

$$\circlearrowleft \cdot \circlearrowleft \approx \circlearrowleft$$

$$f(x_1, \ldots, x_n) \cdot f(y_1, \ldots, y_n) \approx f(x_1 \cdot y_1, \ldots, x_1 \cdot y_1)$$

(ii) For any term s its loop $\circlearrowright_s$ is defined as $s : s \geq s$.

(iii) The residual operation $/$ is defined by, for $\varrho : l \to r \in R$,

$$\begin{aligned}
f(\phi_1,\dots,\phi_n)/f(\psi_1,\dots,\psi_n) &= f(\phi_1/\psi_1,\dots,\phi_n/\psi_n)\\
\varrho(\phi_1,\dots,\phi_n)/l(\psi_1,\dots,\psi_n) &= \varrho(\phi_1/\psi_1,\dots,\phi_n/\psi_n)\\
l(\phi_1,\dots,\phi_n)/\varrho(\psi_1,\dots,\psi_n) &= r(\phi_1/\psi_1,\dots,\phi_n/\psi_n)\\
\varrho(\phi_1,\dots,\phi_n)/\varrho(\psi_1,\dots,\psi_n) &= r(\phi_1/\psi_1,\dots,\phi_n/\psi_n)\\
\chi/(\phi\cdot\psi) &= (\chi/\phi)/\psi\\
(\phi\cdot\psi)/\chi &= \phi/\chi\cdot\psi/(\chi/\phi)\\
\phi/\psi &= \mathsf{tgt}(\phi) \to_{\#} \# \quad \text{otherwise}
\end{aligned}$$

where the 'otherwise' clause applies only if none of the other rules does.

We will often write $\mathcal{T}$ instead of $\hat{\mathcal{T}}$. Notions for residual systems are defined for term rewrite systems via their associated proof term residual system. Note that the defining clauses all have to be taken up to the identities in (i).

Example 6.10 Consider the (parallel) reductions $\mathcal{R}$:$f(\underline{a}) \mathrel{+\!\!\!\!\!\longrightarrow} \underline{f(b)} \mathrel{+\!\!\!\!\!\longrightarrow} g(b,b)$ and $\mathcal{S} : \underline{f(a)} \mathrel{+\!\!\!\!\!\longrightarrow} g(\underline{a},\underline{a}) \mathrel{+\!\!\!\!\!\longrightarrow} g(b,b)$ in the TRS with rules $\varrho : a \to b$ and $\vartheta : f(x) \to g(x,x)$. To check that they are projection equivalent in the proof term residual system, we need to check that the residual of either witness after the other is a loop, i.e. the empty reduction. Let's compute the residual of the first witness $f(\varrho)\cdot\vartheta(b)$ after the second one $\vartheta(a)\cdot g(\varrho,\varrho)$:

$$\begin{aligned}
&\underline{(f(\varrho)\cdot\vartheta(b))/(\vartheta(a)\cdot g(\varrho,\varrho))}\\
&= \underline{f(\varrho)/(\vartheta(a)\cdot g(\varrho,\varrho))}\cdot(\vartheta(b)/\underline{(\vartheta(a)\cdot g(\varrho,\varrho))/f(\varrho)})\\
&= (\underline{f(\varrho)/\vartheta(a)}/g(\varrho,\varrho))\cdot\underline{\vartheta(b)/(\underline{\vartheta(a)/f(\varrho)}\cdot(g(\varrho,\varrho)/\underline{f(\varrho)/\vartheta(a)}))}\\
&= \underline{g(\underline{\varrho/a},\underline{\varrho/a})/g(\varrho,\varrho)}\cdot(\underline{\vartheta(b)/\vartheta(\underline{a/\varrho})}/\underline{g(\varrho,\varrho)/g(\underline{\varrho/a},\underline{\varrho/a})})\\
&= g(\underline{\varrho/\varrho},\underline{\varrho/\varrho})\cdot\underline{g(\underline{b/b},\underline{b/b})/g(\underline{\varrho/\varrho},\underline{\varrho/\varrho})}\\
&= g(b,b)\cdot g(\underline{b/b},\underline{b/b})\\
&= \circlearrowright_{g(b,b)}
\end{aligned}$$

From the residual of either proof term after the other being the loop, one concludes that the reductions $\mathcal{R}$ and $\mathcal{S}$ are projection equivalent.

Introducing an error rule is a radical solution to solve conflicts (critical pairs). Overlapping steps are viewed as being incompatible, hence their only common reduct is the error symbol. However, the following example shows that error symbols may disappear as well.

Example 6.11 Consider the TRS with rules $\{a \to b, a \to c, f(x) \to d\}$. The residuals of $\phi : f(a) \to f(b)$ after $\psi : f(a) \to f(c)$ and vice versa are

$\phi/\psi : f(c) \to f(\#)$ and $\psi/\phi : f(b) \to f(\#)$, respectively. However, extending both ϕ and ψ by a step toward d, by an application of the third rule, yields as residual a loop step for d in both cases. Note how the error symbol has vanished from the residuals. This is caused by erasingness of the third rule $f(x) \to d$.

Theorem 6.12 *$\hat{\mathcal{T}}$ is a residual system, if $\mathcal{T}$ is a left-linear term rewrite system.*

Proof. Our proof proceeds by orienting the identities (i) and rules (iii) of Definition 6.9 into a term rewrite system $\Rightarrow$, and showing that it has the necessary properties, i.e. that it computes a function (is locally confluent and terminating), and satisfies the residual identities.

Local confluence of this TRS is non-trivial because of critical-pairs, e.g.

$$((x \cdot y)/z)/w \Leftarrow (x \cdot y)/(z \cdot w) \Rightarrow (x/(z \cdot w)) \cdot (y/((z \cdot w)/x))$$

One may verify that this critical pair can indeed be completed.

Termination is also non-trivial because of rules nesting / on their right-hand sides, such as $(\phi \cdot \psi)/\chi \Rightarrow \phi/\chi \cdot \psi/(\chi/\phi)$. Our termination proof exploits the, intuitively obvious, fact that the residual of ϕ after ψ is 'smaller' than ϕ. $\square$

Many algebraic laws hold in this residual system, as shown in Table 3. [10] where binary join $\phi \sqcup \psi$ is defined as $\phi \cdot (\psi/\phi)$.

$\phi/\phi = \circlearrowright$	$(\phi/\psi)/(\chi/\psi) = (\phi/\chi)/(\psi/\chi)$
$\phi/\circlearrowright = \phi$	$\chi/(\phi \cdot \psi) = (\chi/\phi)/\psi$
$\circlearrowright/\phi = \circlearrowright$	$(\phi \cdot \psi)/\chi = (\phi/\chi) \cdot (\psi/(\chi/\phi))$
$\phi \cdot \circlearrowright \simeq \phi$	$(\phi \cdot \psi) \cdot \chi \simeq \phi \cdot (\psi \cdot \chi)$
$\circlearrowright \cdot \phi \simeq \phi$	$\phi \sqcup \psi = \phi \cdot (\psi/\phi)$
$\circlearrowright \sqcup \phi \simeq \phi$	$\chi/(\phi \sqcup \psi) = (\chi/\phi)/(\psi/\phi)$
$\phi \sqcup \phi \simeq \phi$	$(\phi \sqcup \psi)/\chi = (\phi/\chi) \sqcup (\psi/\chi)$
$\phi \sqcup \psi \simeq \psi \sqcup \phi$	$\phi \cdot (\psi \sqcup \chi) \simeq (\phi \cdot \psi) \sqcup (\phi \cdot \chi)$
$(\phi \sqcup \psi) \sqcup \chi \simeq \phi \sqcup (\psi \sqcup \chi)$	$\phi \cdot \psi \simeq \phi \cdot \chi \Rightarrow \psi \simeq \chi$

Table 3
Laws for residual systems

Theorem 6.13 *For left-linear TRSs*

(i) *Projection equivalence $\simeq$ is a congruence.*

[10] In fact, these laws hold in any residual system with composition as defined in [47].

(ii) *The projection order $\lesssim$ is a quasi-order.*

(iii) *If $\phi \simeq \psi$, then* $\mathsf{src}(\phi) = \mathsf{src}(\psi)$ *and* $\mathsf{tgt}(\phi) = \mathsf{tgt}(\psi)$*, and similarly for reductions (i.e. modulo the reduction identities).*

Proof. That projection equivalence is an equivalence relation and that the projection order is a quasi-order follows from Lemma 6.8. That projection equivalence is a congruence follows from the defining identities for a term residual system, and from Theorem 6.12 showing that residuals can be computed by orienting these identities into a TRS. The third item is trivial. □

Remark 6.14 (i) Church and Rosser showed in [6] that λ-calculus with β-reduction is confluent. Their proof is based on a construction, projection, for finding the common reduct of two diverging reductions. The construction is based on the observation that a common reduct of two diverging *steps* can be found by contracting the *set* of residuals of either step after the other. Here, contracting a set of redexes is effectuated by selecting some redex in the set for contraction, and repeating this on the set of residuals of the set after this step, until the set is empty. The resulting reduction is called a *complete development* of the set.

(ii) Newman showed in [38] the confluence property for abstract rewrite systems equipped with a projection operation $|$, assigning to every pair ϕ, ψ of co-initial steps a finite set $\phi \mid \psi$ of steps from the target of ψ, which he called the *ψ-derivate* of ϕ. In order to guarantee confluence he assumed, just as we have done above, several axioms on the residual operation. In particular, he assumed the existence of a binary 'disjointness' relation J on co-initial steps satisfying the following two axioms.

(J_1) If $\phi\ J\ \psi$, then $\phi \mid \psi$ has precisely one member.

(J_2) If $\psi_1 \in \phi_1 \mid \chi$ and $\psi_1 \in \phi_1 \mid \chi$, and if $\phi_1\ J\ \phi_2$ or $\phi_1 = \phi_2$, then $\psi_1\ J\ \psi_2$ or $\psi_1 = \psi_2$.

Unfortunately, these axioms failed to hold for his intended application, the λI-calculus. The following counterexample, presented in [43], was discovered by Schroer. [11]

Example 6.15 Let $\omega = \lambda x.xx$ and consider the β-reduction:

$$\omega(\lambda y.\overline{\omega y}) \to (\lambda y.\overline{\omega y})\lambda y.\overline{\omega y} \to \overline{\omega(\lambda y.\overline{\omega y})} \to (\lambda y.\overline{\omega y})\lambda y.\overline{\omega y}$$

where we have overlined the redexes to be discussed below.

(1) By axiom (J_2) we must have that the two residuals of ωy after the first step must be mutually J-related.

(2) After the second step, the whole term and the ωy-redex must be mutually J-related, again by axiom (J_2).

[11] The proof on [38] p. 240 that the axioms do apply, suffers from a (fatal) imprecision in that it is not clear what is meant by variables which are free in two occurrences of a redex, being 'the same'.

(3) Now in the third step, the ωy-redex is duplicated by contracting the whole term and this would lead to a violation of axiom (J_1).

Hindley obtained in [14] the first axiomatic proof of confluence, by then called the Church–Rosser property (CR), which did apply to the λ-calculus. Further efforts on the, rather involved, axiomatic approach seem to have been frustrated by the discovery of the simple and elegant inductive proof of CR for the λ-calculus due to Tait and Martin-Löf (see e.g. [3]). Finally, Melliès presented in [30] an axiomatic approach for proving confluence, which applies to the class of combinatory reduction systems ([21]), hence in particular to the λ-calculus. Comparing his axioms to those of Newman, we observe that if not inspired by, his axioms are still quite close in spirit to Newman's. In particular, the rôle played by his *gripping* relation is analogous to that of Newman's relation J (see [39]).

(iii) In our opinion the main difficulties in the axiomatic approach to term rewriting stem from trying to combine *projection* and *non-(right-)linearity* in one axiomatics. We have separated concerns. We have adapted[12] the axiomatisation for (linear) *projection* of Stark ([45]) and capture *non-linearity* by working with multi-steps, which one may think of as representing complete developments of sets of redexes, instead of with ordinary steps.

7 Equivalence of the equivalences

The main result of this paper relates the four notions of equivalence introduced above.

Theorem 7.1 *Permutation equivalence, parallel standardisation equivalence, labelling equivalence, and projection equivalence are equivalent.*

Proof.

(i) To see that permutation equivalence is equivalent to parallel standardisation equivalence, just note that the latter is a completion of the former (in the TRS sense). More formally, first note that both the parallel standardisation rules (as oriented versions of the inner and outer identities) and the structural identities are permutation identities. Vice versa, every permutation identity is either a structural identity or (can be oriented into) a parallel standardisation rule. Hence, the permutation identities generate the same equivalence as the parallel standardisation steps modulo structural equivalence. Since parallel standard reductions are unique in their equivalence classes by Lemma 4.10, the result follows.

[12] In the axiomatics of Stark ([45]), the projection order is required to be a partial order. However, in contradiction to what is stated there, the projection order is not a partial order in the case of the λ-calculus. For instance, the multi-step contracting both redexes in $(\lambda x.y)(\omega\omega)$ is distinct from, but projection equivalent to the multi-step contracting only the leftmost–outermost redex.

(ii) To see that permutation equivalence implies projection equivalence, note that since projection equivalence is a congruence by Theorem 6.13, it suffices to verify that the left- and right-hand sides of each of the permutation identities are projection equivalent, which is easy. (Note that we do not need the error rule.)

(iii) That projection equivalene implies permutation equivalence, can be seen by a proof by contradiction: assume that $\phi \simeq \psi$ for two proof terms ϕ and ψ between the same two terms, but that ϕ is not permutation equivalent to ψ. By the above, we may therefore assume that and ϕ and ψ are distinct parallel standard reductions, i.e. distinct reductions consisting of parallel steps. Hence we may without loss of generality assume that $\phi = \mathcal{P} \cdot \Phi \cdot \mathcal{R}$ and $\psi = \mathcal{P} \cdot \Psi \cdot \mathcal{S}$ such that Φ and Ψ are distinct parallel steps and $\phi' \in \Phi$ is an ordinary step in the difference between Φ and Ψ which is outermost among the steps in both Φ and Ψ. Roughly speaking, a contradiction now can be derived from the fact that, by its choice, the projection of ϕ cannot be a loop-step, whereas this is required by projection equivalence (see Figure 18).

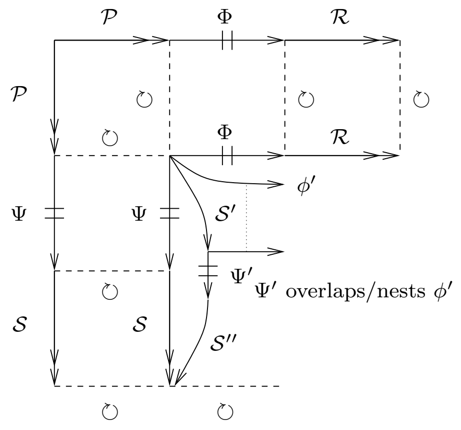

Fig. 18. Projection equivalence implies permutation equivalence

(iv) To see that permutation equivalence implies labelling equivalence, note that the labelling of the targets of the left- and right-hand side of any permutation identity are the same, for a given labelling of their source. Hence, by the inductive definition of proof term labelling, the labelling of the target of a reduction is invariant under permutation equivalence.

(v) To see that labelling equivalence implies permutation equivalence, first note that labelling equivalence implies Lévy labelling equivalence by Theorem 5.23. To show that Lévy labelling implies permutation equivalence is much harder. The proof of this fact in [47] is based on the *Reconstruction* Theorem, expressing that any reduction can be reconstructed, up to permutation equivalence, from the target of its Lévy labelling. That is, Lévy labelled terms have perfect recall, up to permutation equivalence. In the case of our running example, Example 5.2, the reconstruction function will in fact reconstruct $\mathcal{S}$ from the Lévy labelled targets $c^{g^{f(x,b^a)}(b^{\tilde{a}})}$ of both $\mathfrak{L}(\mathcal{R})$ and $\mathfrak{L}(\mathcal{S})$. (Note that the labelling of Example 5.1 does not have the reconstruction property, since we have no way of reconstructing say the intermediate stack $5(1(\top_2))$ from the final labelled term $5(3(\top_2))$.)

The reconstruction property for the unwinding of ARSs is well-known and just expresses that the unwinding allows for perfect recall.

Theorem 7.2 (ARS reconstruction) *Let $\to$ be an abstract rewrite system. Then there exists a reconstruction map $\mathfrak{R}$ from unwound objects to ordinary reductions such that for any reduction $\mathcal{R}$ and any object a' in the unwinding*

$$\mathfrak{R}(a') = \mathcal{R} \textit{ iff } a' = \mathsf{tgt} \circ \mathfrak{U}(\mathcal{R})$$

Lévy labelling is to TRSs as the unwinding is to ARSs, but only allows for perfect recall up to permutation equivalence.

Theorem 7.3 (TRS reconstruction) *Let $\mathcal{T}$ be a left-linear term rewrite system. There exists a reconstruction function $\mathfrak{R}$ which has the property that for any extracted reduction $\mathcal{R}$ and any Lévy labelling s' of its target*

$$\mathfrak{R}(s') = \mathcal{R} \textit{ iff } s' = \mathsf{tgt} \circ \mathfrak{L}(\mathcal{R})$$

Here $\mathcal{R}$ is *extracted*, if $\mathcal{R}$ is parallel standard and all its steps *trace* to its final term s. We will not formally define the notion of tracing here, and refer the reader to [47] for details, but the rough idea is as follows. Given a term rewrite step $C[l^\sigma] \to C[r^\sigma]$, all positions in the context C and substitution part σ trace to 'themselves' and all positions in the pattern l of the left-hand side trace to all positions in the pattern r of the right-hand side.

□

As an easy corollary to the theorem we have that the permutation order and the projection order are equivalent, as orders on ordinary proof terms. However, when transferring properties from the projection order to the permutation order, one should keep in mind that only the proof terms of the former may contain error rule symbols and that it may be because of these that some property, e.g. having upper bounds, of the projection is brought about. In the case of orthogonal rewriting systems error rules are not necessary.

Proposition 7.4 *Let $\hat{\mathcal{T}}$ be the proof term residual system of the orthogonal term rewriting system $\mathcal{T}$. Then, the 'otherwise'-clause in Definition 6.9 never applies when computing residuals.*

This can be used to answer the following question posed by Newman in [38]: [13]

> The results are essentially about "partially-ordered" systems, i.e. sets in which there is a transitive relation $>$, and sufficient conditions are given for every two elements to have a lower bound (i.e. for the set to be "directed") if it is known that every two "sufficiently near" elements have a lower bound.

[13] As far as we know, the 'later discussion' mentioned never took place.

> What further questions are required for the existence of a *greatest* lower bound is not relevant to the present purpose, and is reserved for a later discussion.

The results of this paper provide a solution of sorts to this question (see [35] for a categorical approach).

Theorem 7.5 *The converse $\sqsupseteq$ of the permutation order of an orthogonal TRS has greatest lower bounds.*

Proof. The permutation order and the projection order are equivalent for ordinary proof terms. Since the latter has least upper bounds (computed by $\sqcup$), and since these do not contain error symbols by the proposition, the former has least upper bounds as well. Hence its converse has greatest lower bounds. $\square$

In interpreting this result, one should be aware of syntactic accidents. Reconsider the TRS with single rule $\varrho : f(x) \to x$ of Example 2.1. The distinct steps $\varrho(f(a))$ and $f(\varrho(a))$ have the same target $f(a)$. Nevertheless, the residual of either after the other is non-empty:

$$\varrho(f(a))/f(\varrho(a)) = \varrho(f(a)/\varrho(a)) = \varrho(a)$$
$$f(\varrho(a))/\varrho(f(a)) = \varrho(a)/f(a) = \varrho(a)$$

and the least common extension of both of them ends in a rather than in $f(a)$!

Remark 7.6 (i) The correspondence between permutation equivalence and labelling equivalence only holds for non-collapsing term rewrite systems, i.e. term rewrite systems which do not contain rules whose right-hand sides are single variables. This *collapse* problem can be overcome by *expanding* collapsing systems to non-collapsing systems first ([47]). In this way, the correspondence can be made to hold for *all* (left-linear) term rewrite systems.

(ii) Maybe tracing is best formalised by mapping terms to complexes in combinatorial topology.[14] The point is that in present formalisations the context-part of a rewrite step is not really stable during a rewrite step, making tracing cumbersome to express. This seems more natural in topology, where instead of modifying the context to 'make room for a big right-hand side' one may just shrink the right-hand side in order to fit into to the hole left by removing the left-hand side. On the other hand, it is not so clear how

[14] The analogy between rewriting and combinatorial topology was already noted by Newman ([38]):

> In combinatorial topology the objects are complexes, and the allowed moves are "breaking an edge" by the insertion of a new vertex, or the reverse of this process. In Church's "conversion calculus" the rules II and III are "moves of this kind".

We do not know of work exploiting this analogy.

to deal with the non-linear phenomena of erasure and duplication, which are omnipresent in rewriting, in an elegant topological way.

References

[1] Z.M. Ariola and J.W. Klop. Equational term graph rewriting. *Fundamenta Informaticae*, 26:207–240, 1996.

[2] A. Asperti and S. Guerrini. *The Optimal Implementation of Functional Programming Languages*, volume 45 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1998.

[3] H.P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 2nd revised edition, 1984.

[4] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[5] G. Boudol. Computational semantics of term rewriting systems. In M. Nivat and J.C. Reynolds, editors, *Algebraic Methods in Semantics*, pages 169–236. Cambridge University Press, 1985.

[6] A. Church and J.B. Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, 39:472–482, January to June 1936.

[7] H. Cirstea and C. Kirchner. Introduction to the rewriting calculus. Technical Report RR-3818, INRIA, December 1999.

[8] H.B. Curry. A new proof of the Church–Rosser theorem. *Koninklijke Nederlandse Akademie van Wetenschappen, Proceedings, series A*, 55:16–23, 1952.

[9] H.B. Curry and R. Feys. *Combinatory Logic*, volume I. North-Holland, 1958.

[10] F. Gadducci and U. Montanari. Comparing logics for rewriting: rewriting logic, action calculi and tile logic. *Theoretical Computer Science*, 285(2):319–358, 2002.

[11] G. Gonthier, J.-J. Lévy, and P.-A. Melliès. An abstract standardisation theorem. In *Proceedings of the Seventh Annual IEEE Symposium on Logic in Computer Science (LICS '92)*, pages 72–81. IEEE Computer Society Press, 1992.

[12] J.N. Gray and A. Reuter. *Transaction Processing Concepts and Techniques.* Morgan Kaufmann Publishers, 1993.

[13] B.P. Hilken. Towards a proof theory of rewriting: The simply typed 2λ-calculus. *Theoretical Computer Science*, 170(1–2):407–444, 1996.

[14] J.R. Hindley. An abstract form of the Church–Rosser theorem I. *the Journal of Symbolic Logic*, 34(4):545–560, December 1969.

[15] J.R. Hindley. Standard and normal reductions. *Transactions of the American Mathematical Society*, 241:253–271, July 1978.

[16] G. Huet and J.-J. Lévy. Computations in orthogonal rewriting systems, part I + II. In J.L. Lassez and G.D. Plotkin, editors, *Computational Logic – Essays in Honor of Alan Robinson*, chapter 11 + 12, pages 395–443. MIT Press, 1991. Update of: Call-by-need computations in non-ambiguous linear term rewriting systems, 1979.

[17] F. Joachimski. *Reduction Properties of* Π*IE-Systems.* PhD thesis, Ludwig-Maximilians-Universität München, 2001.

[18] Z. Khasidashvili. Expression reduction systems. *Proceedings of I. Vekua Institute of Applied Mathematics*, 36:200–220, 1990.

[19] Z. Khasidashvili and J.R.W. Glauert. Discrete normalization and standardization in deterministic residual structures. In M. Hanus and M. Rodríguez-Artalejo, editors, *Proceedings of the Fifth International Conference on Algebraic and Logic Programming, (ALP '96)*, volume 1139 of *Lecture Notes in Computer Science*, pages 135–149. Springer, 1996.

[20] Z. Khasidashvili and J.R.W. Glauert. Relating conflict-free stable transition and event models via redex families. *Theoretical Computer Science*, 286(1):65–95, 2002.

[21] J.W. Klop. *Combinatory Reduction Systems.* PhD thesis, Rijksuniversiteit Utrecht, June 1980.

[22] J.W. Klop. Term rewriting systems. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 1–116. Oxford University Press, New York, 1992.

[23] C. Laneve and U. Montanari. Axiomatizing permutation equivalence. *Mathematical Structures in Computer Science*, 6:219–249, 1996.

[24] J.-J. Lévy. *Réductions correctes et optimales dans le* λ*-calcul.* Thèse de doctorat d'état, Université Paris VII, 1978.

[25] S. Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics.* Springer, 2nd edition, December 1998.

[26] L. Maranget. Optimal derivations in orthogonal term rewriting systems and in weak lambda calculi. In *Conference Record of the Eighteenth Annual ACM Symposium on Principles of Programming Languages (POPL '91)*, pages 255–269. ACM Press, 1991.

[27] L. Maranget. *La stratégie paresseuse.* Thèse de doctorat, Université Paris VII, 6 Juilliet 1992.

[28] N. Martí-Oliet and J. Meseguer. Rewriting logic: Roadmap and bibliography. *Theoretical Computer Science*, 285(2):121–154, 2002.

[29] A. Mazurkiewicz. Concurrent program schemes and their interpretation. Technical Report DAIMI PB-78, Aarhus University, 1977.

[30] P.-A. Melliès. *Description Abstraite des Systèmes de Réécriture.* Thèse de doctorat, Université Paris VII, 20 Décembre 1996.

[31] P.-A. Melliès. Axiomatic rewriting theory III, a factorisation theorem in rewriting theory. In E. Moggi and G. Rosolini, editors, *Proceedings of the Seventh International Conference on Category Theory and Computer Science (CTCS '97)*, volume 1290 of *Lecture Notes in Computer Science*, pages 49–68. Springer, 1997.

[32] P.-A. Melliès. Axiomatic rewriting theory IV, a stability theorem in rewriting theory. In *Proceedings of the Thirteenth Annual IEEE Symposium on Logic in Computer Science (LICS '98)*, pages 287–298. IEEE Computer Society Press, 1998.

[33] P.-A. Melliès. Axiomatic rewriting theory II, the $\lambda\sigma$-calculus enjoys finite normalisation cones. *Journal of Logic and Computation*, 10(3):461–487, 2000.

[34] P.-A. Melliès. Axiomatic rewriting theory I, a diagrammatic standardization theorem, 2002. Submitted. 71 pp.

[35] P.-A. Melliès. Axiomatic rewriting theory VI, residual theory revisited. In S. Tison, editor, *Proceedings of the Thirteenth International Conference on Rewriting Techniques and Applications (RTA '02)*, volume 2378 of *Lecture Notes in Computer Science*, pages 24–50. Springer, 2002.

[36] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96:73–155, 1992.

[37] R. Milner. *A Calculus of Communicating Systems.* Springer, Berlin, 1980.

[38] M.H.A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43(2):223–243, April 1942.

[39] V. van Oostrom. FD à la Melliès. Typescript, 4 pp., 1997.

[40] V. van Oostrom. Finite family developments. In H. Comon, editor, *Proceedings of the Eighth International Conference on Rewriting Techniques and Applications (RTA '97)*, volume 1232 of *Lecture Notes in Computer Science*, pages 308–322. Springer, June 1997.

[41] V. van Oostrom. Normalisation in weakly orthogonal rewriting. In P. Narendran and M. Rusinowitch, editors, *Proceedings of the Tenth International Conference on Rewriting Techniques and Applications (RTA '99)*, volume 1631 of *Lecture Notes in Computer Science*, pages 60–74. Springer, 1999.

[42] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings of the Fifth GI Conference on Theoretical Computer Science, Karlsruhe*, Lecture Notes in Computer Science, pages 167–183. Springer, 1981.

[43] J.B. Rosser. Review of [8]. *the Journal of Symbolic Logic*, 21(4):337–426, April 1956.

[44] D.E. Schroer. *The Church–Rosser Theorem.* PhD thesis, Cornell University, 1965.

[45] E.W. Stark. Concurrent transition systems. *Theoretical Computer Science*, 64:221–269, 1989.

[46] Y. Sun. Term rewriting and Hoare logic – coded rewriting. *Information Processing Letters*, 60(5):237–242, 1996.

[47] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

[48] J.S. Vitter and P. Flajolet. Average-case analysis of algorithms and data structures. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, Algorithms and Complexity, pages 431–524. Elsevier, Amsterdam, 1990.

[49] G. Winskel and M. Nielsen. Models for concurrency. In S. Abramsky, D. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 4, pages 1–148. Oxford University Press, 1995.