On Causal Equivalence in String Rewriting

Vincent van Oostrom[*]
oostrom@javakade.nl

We introduce proof terms for string rewrite systems and, using these, show that various notions of equivalence on reductions known from the literature can be viewed as different perspectives on the notion of causal equivalence. In particular, we show that *permutation equivalence* classes (as known from the λ-calculus and term rewriting) are uniquely represented both by *trace graphs* (known from physics as causal graphs) and by so-called *greedy-multistep* reductions (as known from algebra). We present effective maps from the former to the latter, *topological multi-sorting* TS, and vice versa, the *proof term algebra* $[\![\,]\!]$.

1 Introduction

In general, we are interested in all aspects of computations as modelled by rewrite systems. Here, we are interested in *finite* computations 'doing the same work up to the order of the tasks performed'. This can be analysed from the perspective of *causality* with the idea that it is exactly the causally independent tasks that can be reordered. Given that causality is omnipresent, it is no surprise it has been discussed and mathematically modelled in many ways; to mention a few [21, 17, 4, 25, 28, 31, 11, 10, 20, 16, 18, 13, 8, 9, 7, 33]. In [30, Chapter 8] we showed the various notions of equivalence of reductions as known from the λ-calculus and term rewriting literature to be the same in the case of left-linear *term* rewrite systems (TRSs).[1] The goal of this paper is to do the same for *string* rewrite systems, guided by that strings can be represented as terms so that extant theory for term rewriting can be adapted to string rewriting.

String rewriting affords better properties than term rewriting due to linearity: whereas term rewrite steps may be *non-linear* as they can *replicate*, erase or copy, subterms of arbitrary sizes, string rewrite steps cannot do so; they are *linear*. We moreover forbid left- and right-hand sides of rules to be the empty string. This restriction makes sense from the perspective of causality [31] as it entails all steps being *ex materia* (forbidding *ex nihilo* steps) and having *finite* causes (forbidding *infinite* causes). By imposing these (linearity and non-emptiness) restrictions, we are in a sweet spot; the resulting string rewrite systems have *sufficient* structure to *express* the different perspectives on causal equivalence, and these perspectives can in turn be proven equivalent with in a *simple* way due to the absence of replication. As in [30, Chapter 8], to state and prove results *proof* terms are our tool of choice.

2 Proof terms for string rewriting

The usual definition of the finite strings over an alphabet Σ as the free monoid over Σ is abstract. To be able to deal with matters of representation, we instead will be concrete here.

[*]Supported by EPSRC Project EP/R029121/1 Typed lambda-calculi with sharing and unsharing. Most of this work was performed while employed at the University of Bath, England.

[1]This was (partially) extended from first- to higher-order term rewriting in [5]; cf. [2] for a recent reprise of the orthogonal higher-order case.

Submitted to:
TERMGRAPH 2022

Definition 1. *A term rewrite system is* oudenadic *if all rule and function symbols have arity* 0*, it has a nullary symbol ε (empty string) and a binary composition symbol $\cdot_h$, and terms are considered modulo $\equiv_M$ induced by the monoid laws, i.e. $\varepsilon \cdot_h s = s$, $s \cdot_h \varepsilon = s$, and $(s \cdot_h t) \cdot_h u = s \cdot_h (t \cdot_h u)$.*

We assume $\cdot_h$ is infix and right-associative and that it is left implicit, i.e. is represented by juxtaposition. To that end, we assume Σ has unique reading.

Remark 1. *In our modelling strings are closed oudenadic terms over the alphabet modulo the monoid laws. Uniquely representing such equivalence classes can itself be achieved by term rewriting: orienting the above monoid laws from left to right yields a complete (confluent and terminating) term rewrite system, having as normal forms strings of shape either ε or $a_1 \dots a_n$ for some $n \geq 1$.*

Example 1. *The alphabet $\Sigma := \{A, B\}$ has unique reading. Per our conventions $ABAAB$ abbreviates the term $A \cdot_h (B \cdot_h (A \cdot_h (A \cdot_h B)))$, which is closed and in normal form with respect to the monoid rules, so serves as the unique representative of the string (an $\equiv_M$-equivalence class containing, e.g., $(AB)(AA)B$).*

Concretely, a *string* rewrite system *over* an alphabet Σ is an oudenadic term rewrite system having the letters in Σ as nullary function symbols, with sources and targets of rules being nonempty strings (cf. the introduction), and steps taking place modulo $\equiv_M$.

Remark 2. *String rewriting could alternatively be represented by means of* monadic *term rewriting, associating a* unary *function symbol to each letter; cf. [30, Section 3.4.4] for an account of both representations. Our terminology* oudenadic *is an attempt to high-light that the representation employed here associates* nullary *function symbols to letters.*

We consider term rewrite systems in the sense of [30, Chapters 8 and 9], meaning that rules themselves will feature as *symbols* whose arity (0 for the oudenadic systems we consider here) is the number of variables in the rule, and rules come equipped with source / target functions mapping them to their lhs / rhs. This enables expressing reductions, and more generally proofs in rewrite logic [20], as *proof* terms [30], terms over a signature comprising the letters, the rules, and a binary composition symbol $\cdot_v$ representing the transitivity inference rule of rewrite logic [20]. In turn, this allows us to represent the key notion of this paper, the notion of causal equivalence, as an equivalence on reductions / proof *terms*.

Definition 2. *Consider for an oudenadic term rewrite system $\langle \Sigma, P \rangle$, the extension of the oudenadic signature for Σ by the rules ρ in P as nullary symbols and the binary composition symbol $\cdot_v$.* Proof *terms are a subset of the terms over this signature defined inductively, together with* source *and* target *functions* src *and* tgt *to strings, on the left in Table 2, where we use $\gamma : s \geqslant t$ to denote that γ is a proof term having string s as source and string t as target, and employ $\gamma, \delta, \zeta, \eta, \dots$ to range over proof terms.*

(empty)	$\varepsilon : \quad \varepsilon \geqslant \varepsilon$	
(letter)	$a : \quad a \geqslant a$	for each letter a
(rule)	$\rho : \quad \ell \geqslant r$	for each rule $\rho : \ell \to r$
(juxtaposition)	$\gamma_1\gamma_2 : s_1 s_2 \geqslant t_1 t_2$	if $\gamma_i : s_i \geqslant t_i$
(transitivity)	$\gamma \cdot \delta : \quad s \geqslant u$	if $\gamma : s \geqslant t$, and $\delta : t \geqslant u$

Table 1: Proof terms for string rewriting

Remark 3.
- *To keep the two binary compositions $\cdot_h$ and $\cdot_v$ apart, we refer to the former as* horizontal *and to the latter as* vertical *composition. We abbreviate vertical composition $\cdot_v$ to $\cdot$, assume it is right-associative, and that it binds weaker than horizontal composition $\cdot_h$ / juxtaposition.*

(h-left unit)	$\varepsilon\gamma = \gamma$	(v-left unit)	$s \cdot \gamma = \gamma$
(h-right unit)	$\gamma\varepsilon = \gamma$	(v-right unit)	$\gamma \cdot t = \gamma$
(h-associativity)	$(\gamma\delta)\zeta = \gamma(\delta\zeta)$	(v-associativity)	$(\gamma\cdot\delta)\cdot\zeta = \gamma\cdot(\delta\cdot\zeta)$
(exchange)	$\gamma\delta\cdot\zeta\eta = (\gamma\cdot\zeta)(\delta\cdot\eta)$		

Table 2: Laws generating permutation equivalence

- *By the vertical composition being on* strings *the target of γ is only required to be equivalent modulo the monoid laws to the source of δ in (transitivity). We have $t : t \geqslant t$ for every oudenadic term t.*

The name proof term for such terms is justified by that they be viewed as a *proof* that their target string is *reachable* from their source string by using the rewrite rules. Building on Example 1, we take the following as a running example to illustrate concepts and results.

Example 2. *Let $\langle \Sigma, P\rangle$ be the string rewrite system having rules $P := \{\alpha : BB \to A, \beta : AAB \to BAAB\}$. The proof term $\gamma := AB\beta \cdot A\alpha AAB \cdot AA\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha AABAAB \cdot A\beta AAB$ proves the* reachability *statement $ABAAB \geqslant ABAABAAB$. An alternative witness to that statement is the proof term $\gamma' := AB\beta \cdot A\alpha\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha\beta AAB$.*

For the (vertical) compositions in these proof terms to be well-defined it is essential to work modulo the monoid laws. For instance, although the target $(BAAB)AAB$ of βAAB and the source $B(AAB)AAB$ of $B\beta AAB$ are distinct as oudenadic terms, they are both represented by the string $BAABAAB$, allowing their vertical composition in γ.

Although in the example the proof terms γ and γ' intuitively do 'the same amount of work', the latter is shorter than the former. This is due to that the former is maximally sequentialised, performing one step at the time, whereas the latter is maximally concurrent, performing steps as soon as possible as concurrency permits.

Definition 3. *A* multistep *is a proof term without vertical compositions. It is* empty */ a (*single*)* step *if it has no / one occurrence of a rule. A (multistep)* reduction *either is an empty multistep or a vertical composition, associated to the right, of nonempty (multi)steps.* Permutation *equivalence $\equiv$ between proof terms is generated by the equivalences in Table 2, where the sides of the equivalences are restricted to proof terms, i.e. sources and targets of the proof terms γ, δ, ζ and the oudenadic term s are assumed to match appropriately.*

We use $\Phi, \Psi, X, \ldots$ to range over multisteps, and $\phi, \psi, \chi, \ldots$ to range over steps. Observe that the source / target of the left- and right-hand side of each law in Table 2 are the same (as strings).

Remark 4. *Our reductions, as proof terms of a specific shape, are formally distinct from the classical notion of a reduction, as a finite sequence of steps, in rewriting [1, 30]. However, since there is an obvious bijection between both we feel the confusion is acceptable. For instance, the proof term $\gamma := AB\beta \cdot A\alpha AAB \cdot AA\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha AABAAB \cdot A\beta AAB : ABAAB \geqslant ABAABAAB$ corresponds to:*

$$AB\underline{AAB} \to A\underline{BB}AAB \to AA\underline{AAB} \to \underline{AAB}AAB \to B\underline{AAB}AAB \to \underline{BB}AABAAB \to A\underline{AAB}AAB \to ABAABAAB$$

Similarly, the proof term $\gamma' := AB\beta \cdot A\alpha\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha\beta AAB$ corresponds to the following sequence of multisteps, where we employ the notation $\mathrel{-\!\!\circ\!\!\!\rightarrow}$ of [30, Chapter 8] for multisteps:

$$AB\underline{AAB} \mathrel{-\!\!\circ\!\!\!\rightarrow} A\underline{BB}\,\underline{AAB} \mathrel{-\!\!\circ\!\!\!\rightarrow} \underline{AAB}AAB \mathrel{-\!\!\circ\!\!\!\rightarrow} B\underline{AAB}AAB \mathrel{-\!\!\circ\!\!\!\rightarrow} \underline{BB}\,\underline{AAB}AAB \mathrel{-\!\!\circ\!\!\!\rightarrow} ABAABAAB$$

Logicality, cf. [22], of reductions expresses that if a reachability statement holds then it is provable by a reduction that is permutation equivalent to the original proof term.

Lemma 1 (Logicality)**.** *If $\gamma : s \geqslant t$ for some proof term γ, then there is a reduction $\gamma' : s \geqslant t$ with $\gamma \equiv \gamma'$.*

Proof. By induction and cases on γ.

(empty) the empty string ε is an empty reduction;

(letter) a single letter a is an empty reduction;

(rule) a single rule ρ is a single step reduction from its lhs to its rhs;

(juxtaposition) suppose to have a proof term $\gamma := \gamma_1\gamma_2 : s_1s_2 \geqslant t_1t_2$ with $\gamma_i : s_i \geqslant t_i$. By the IH we have reductions $\gamma_i' : s_i \geqslant t_i$ with $\gamma_i \equiv \gamma_i'$. Set γ' to $\gamma_1'\langle s_2\rangle \cdot \langle t_1\rangle\gamma_2'$, where for a reduction ζ and string u, $\gamma\langle u\rangle$ denotes the reduction obtained by suffixing each step of γ by u, and symmetrically for $\langle u\rangle\gamma$. One verifies $\gamma' : s_1s_2 \geqslant t_1t_2$ and $\gamma \equiv \gamma'$ by using (exchange) and vertical units unit repeatedly. Then by repeated vertical associativity applied to γ' we obtain a reduction, except in case one or both of the γ_i' is the empty reduction in which case we conclude by eliding one such by a horizontal unit.

(transitivity) by vertically composing the reductions obtained by the IH for the constituent proof terms, possibly followed by associating to right and eliding empty reductions as before. □

The proof is effective, transforming proof terms into reductions witnessing the same reachability.

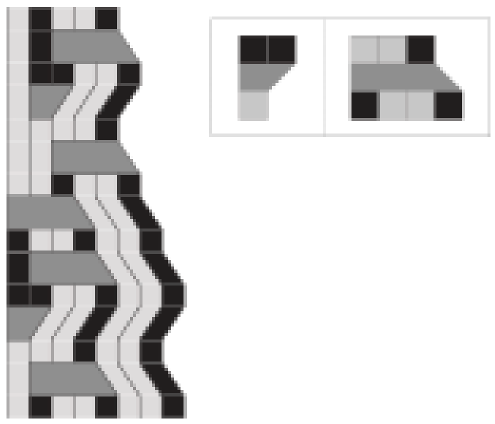
Example 3. *The procedure underlying the proof of Lemma 1 transform the proof term (in fact a multistep reduction) γ' of Example 2 into the reduction γ. To see this it suffices, since vertical compositions transform homomorphically, to note that the multisteps $A\alpha\beta$ and $\alpha\beta AAB$ in γ' are transformed into the (two step) reductions $A\alpha AAB \cdot AA\beta$ and $\alpha AABAAB \cdot A\beta AAB$ in γ, respectively.*

Remark 5. *Logicality is the raison d'être for the field of rewriting [1, 30], allowing to reduce the study of reductions to that of steps. Cf. [20, Lemma 3.6] for the corresponding logicality result for term rewriting.*

Although the logicality lemma allows to represent any proof term by a reduction, the latter is in general far from unique (up to permutation equivalence). For instance, in Example 3 we could have chosen to transform the multistep $A\alpha\beta$ in γ' into the two step reduction $ABB\beta \cdot A\alpha BAAB$ instead, giving rise to a reduction permutation equivalent to γ' but distinct from γ. Intuitively this is caused by that factorising a proof term into a sequence of steps forces to order steps in *some* (arbitrary) way even though they may be causally independent. For instance, α and β in the multistep $A\alpha\beta$ are concurrent / causally independent, but still must be ordered to obtain a reduction; both orders will do. Such a representation favours sequentiality over concurrency and length over width, so to speak. In the next sections we will go into the opposite direction, maximally favouring concurrency over sequentiality and width over length.

From that perspective, the proof term $\gamma := AB\beta \cdot A\alpha AAB \cdot AA\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha AABAAB \cdot A\beta AAB$ is a proof of the reachability statement $ABAAB \geqslant ABAABAAB$ that is wasteful in two ways:

(too long) This can be remedied by proceeding greedily [7], employing proper *multi*steps instead of steps. For instance, the second and third steps $A\alpha AAB \cdot AA\beta : ABBAAB \geqslant AABAAB$ in γ can be combined into the single multistep $A\alpha\beta : ABBAAB \geqslant AABAAB$. Proceeding greedily, combining as many of the single steps into multisteps as possible, and as early as possible, turns γ into the shorter *greedy* multistep reduction $\gamma' := AB\beta \cdot A\alpha\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha\beta AAB$. As we will show, greedy multistep reductions may serve as *unique* representatives of permutation equivalence classes.



$$\begin{tikzcd}[row sep=small]
ABAAB \arrow[d, "AB\beta"] \\
ABBAAB \arrow[d, "A\alpha AAB"] \\
AAAAB \arrow[d, "AA\beta"] \\
AABAAB \arrow[d, "\beta AAB"] \\
BAABAAB \arrow[d, "B\beta AAB"] \\
BBAABAAB \arrow[d, "\alpha AABAAB"] \\
AAABAAB \arrow[d, "A\beta AAB"] \\
ABAABAAB
\end{tikzcd}$$

$$\begin{tikzcd}
BB \arrow[d, "\alpha"] & AAB \arrow[d, "\beta"] \\
A & BAAB
\end{tikzcd}$$

Figure 1: Reduction $\gamma : ABAAB \geqslant ABAABAAB$ (right) and its evolution (left)

(too large) (Multi)steps not only represent what changes, rules, but also what *does not change*, letters (cf. the frame problem). As a consequence, in general proof terms predominantly consist of letters; this holds true in particular both for γ and γ'. *Causal graphs* [32] (cf. Figure 2 left) remedy this by eliding letters, only keeping the causal dependencies between rule symbols. This suffices, as we will show, to let causal graphs serve as *unique* representatives of permutation equivalence classes.

To express and relate both remedies we will employ a bit of *residual* theory (going back to [6]) for multisteps below. To avoid things becoming too heavy for this short paper, we only develop the residual theory necessary here and in an ad hoc informal fashion, referring the reader to Chapter 8 of [30] in general and to Section 8.7 in particular, for background on (from the perspective of permutation equivalence) and a formal treatment of, residuation.

Definition 4. *For multisteps Φ, Ψ having the same source, we write $\Phi \subseteq \Psi$ to denote that Φ is* contained *in Ψ, meaning that Φ is obtained from Ψ by mapping* some *occurrences of rule symbols to their source. In that case, we denote by Ψ/Φ the* residual *of Ψ* after *Φ, that is, the multistep obtained from Ψ by mapping the* other *occurrences of rules (the* complement *of those selected for $\Phi \subseteq \Psi$) to their target.*

Example 4. *$ABBAAB$, $ABB\beta$, $A\alpha AAB$ and $A\alpha\beta$ are the four multisteps contained in $A\alpha\beta$ in Example 2. We have, e.g., $A\alpha\beta/ABB\beta = A\alpha BAAB$ and $A\alpha\beta/A\alpha AAB = AA\beta$. Observe that if $\Phi \subseteq \Psi$ and Φ is nonempty, then fewer rule symbols occur in Ψ/Φ than in Ψ by linearity of string rewriting.*

3 Trace graphs by proof term algebra

We give a proof term algebra $[\![\,]\!]$ into tragrs, trace graphs, based on the causal graphs of [32]. The algebra is shown to model permutation equivalence in that permutation equivalent proof terms are mapped to the same tragr. We give a procedure dubbed topological multisorting, reading back a proof term from a tragr.

Before giving a formal treatment, we first give some underlying intuitions by means of an example that links to the intermediate informal notion of an *evolution* [32], and to our discussion above.

Example 5. *The reduction γ of Example 2 can be depicted as the evolution on the left of in Fig. 1 (taken from [27]; based on [32, fig. a, p.498]). To that end, we interpret steps as rows of (possibly skewed) blocks obtained by $A \mapsto$ □, $B \mapsto$ ■, $\alpha \mapsto$ ▰, and $\beta \mapsto$ ⏢. Vertical compositions of steps are interpreted by stacking the rows of the interpretations of the steps on top of each other, interspersed*

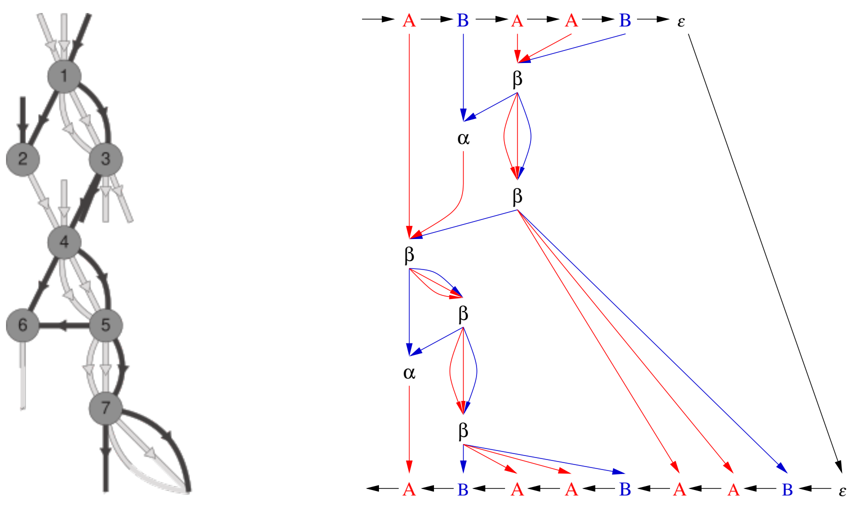


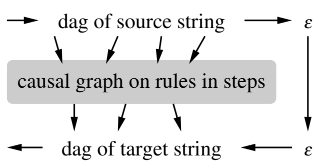
Figure 2: Causal graph (left) and tragr from *ABAAB* to *ABAABAAB* (right)

with the evaluations of their sources and targets. For instance, the top three rows are the evaluations □■□□■, □■▬, *and* □■■□□■ *of the source, step, and target of* $AB\beta : ABAAB \geqslant ABBAAB$*. (The interpretations of the rule symbols* α, β *are given next to the evolution).*

Looking at Figure 1 the correspondence between evolutions and reductions is clear though informal. Evolutions nicely illustrate the point argued above in (too large) that letters (the white and black boxes representing A and B) add nothing to the representation; the source and target strings and the causal dependencies (represented by directed edges) between the rule symbols would suffice to read back the multistep reduction γ' (permutation equivalent to γ) from the evolution. That idea will be formalised below using the notion of *tragr*,[2] short for *trace graph*, illustrated for γ / γ' in Figure 2.

Remark 6. *The book [32] being intended for a general audience, causal graphs are not sufficiently formalised there to state our results here; in particular, causal graphs lack what we call below an interface (dags of the source and target strings). Tragrs are our way to overcome that deficiency. We believe that if Wolfram were to formalise his notion of causal graph, he would end up with something similar to tragrs.*

Definition 5. *Given a string rewrite system* (Σ, P)*, a* tragr from string s to string t *is a port graph [3, 15, 29, 24] comprising the following three parts, as visualised in:*



- *the dag of source string s having for every occurrence of a letter a in s a node labelled a, having (in clockwise order) an input port of type $*$, an output port of type $*$, and an output port of type a. The nodes are connected in a straight line by edges of type $*$, terminated by a node labelled ε having an input port of type $*$, and an output port of type ε.*

[2] Pronounce as *tracker*.

- *a dag, the* causal *graph, of nodes labelled by rule symbols ρ having (in clockwise order) as input ports the letters of the source string of ρ and as output ports the letters of (the reverse of) the target string, with each port having the type of its letter;*
- *the dag of target string t, as for the source string but in reverse direction, i.e. with input and output port of type $*$ swapped.*

The tragr is required to be a planar dag, to only have edges from input to output ports of the same type, and to have exactly two ports without edges, both of type $*$: the first input port of the source string and the first output port of the target string.

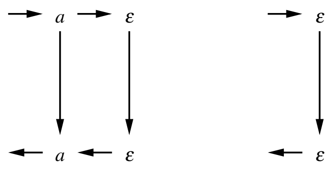
We indicate *the* input / output ports of a tragr by dangling edges, and refer to the dags of the source and target strings combined as its *interface*.

Example 6. *The graph on the right in Figure 2 is a tragr with the types of edges being indicated by color.*

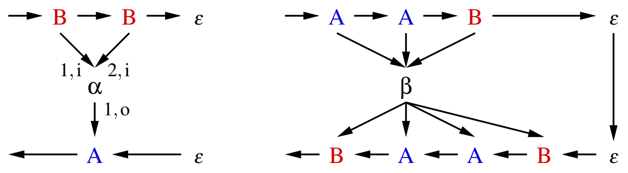
Remark 7. *Tragrs are not (too large) in the sense discussed above; letters only feature in the interface but not in the causal graph of a tragr; cf. the text below [30, Def. 8.6.17].*

Definition 6. *For a string rewrite system (Σ, P) the proof term algebra $[\![\,]\!]$ on tragrs is given by:*

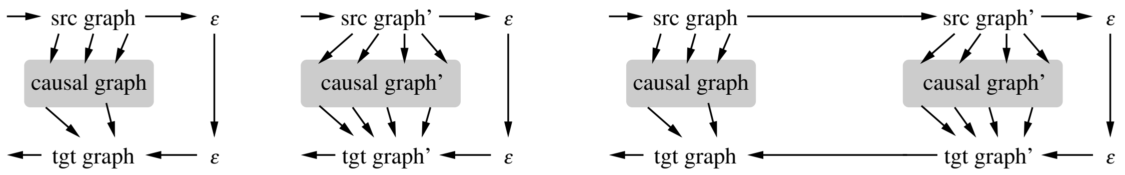
(letter and empty) $[\![a]\!]$ *and* $[\![\varepsilon]\!]$ *are the tragrs:*



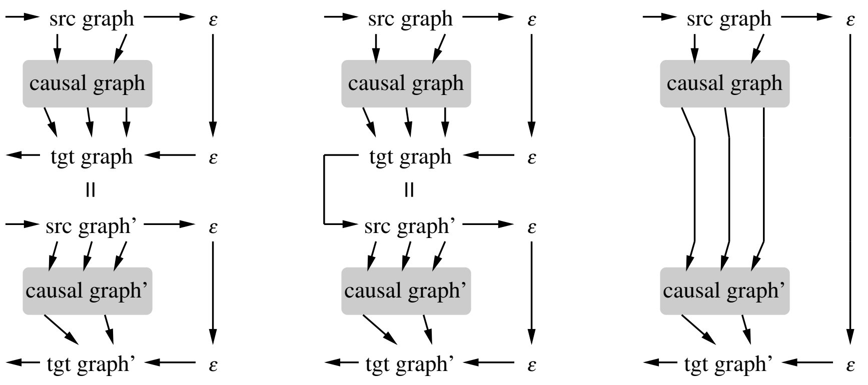
(rule) $[\![\rho]\!]$ *is a tragr having the straight line dags for its source and target as interface, comprising a single rule node connected to the interface in an orderly way, illustrated for rules α and β by:*



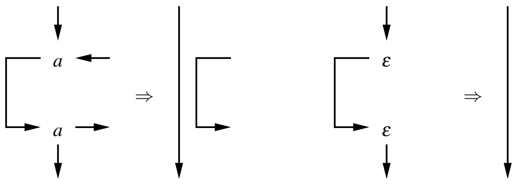
(juxtaposition) $[\![\gamma\delta]\!]$ *is obtained from* $[\![\gamma]\!]$ *and* $[\![\delta]\!]$ *by removing the εs from the former, and redirecting the input and output of the latter accordingly:*



(transitivity) The tragr $[\![\gamma \cdot \delta]\!]$ *is obtained from* $[\![\gamma]\!]$ *and* $[\![\delta]\!]$ *by connecting the output of the former to the input of the latter, and subsequently eliding the intermediate interface:*



where elision from the middle to the right is achieved by normalising with respect to the rules:



Observe that if $\gamma\colon s \geqslant t$ then $[\![\gamma]\!]$ is a tragr from s to t.

Example 7. *The tragrs $[\![\gamma]\!]$ and $[\![\gamma']\!]$ of the permutation equivalent γ,γ' are as on the right in Figure 2.*

Remark 8.
- *Elision $\Rightarrow$ is complete: terminating because the number of nodes decreases in each step and confluent because elision can be viewed as an interaction net rule [15].*
- *$[\![\gamma]\!]$ is finite so that all maximal paths in it lead from its input to its output, using that nodes have at least one input / output port, by the assumption that left- and right-hand sides are non-empty.*
- *We modelled trace graphs, tragrs, after the* trace relations *of [30, Definition 8.6.17 / Figure 8.37] with the main difference between both being that the latter do not allow* parallel *edges between the same two nodes. That makes the latter unsuitable for our purposes here; only knowing* that *a rule causally depends on another not* how, *is in general not sufficient to read back proof terms. For instance, for rules $A \to BBB$, $BB \to B$ and $BB \to C$, the reductions $A \to \underline{BB}B \to BB \to C$ and $A \to B\underline{BB} \to BB \to C$ induce the same trace relation, despite not being permutation equivalent.*[3]
- *The algebra $[\![\,]\!]$ illustrates that horizontal and vertical composition are closely related to parallel and series composition of graphs.*

We show that $[\![\,]\!]$ maps permutation equivalent proof terms to the same tragr,[4] see Example 7, deferring showing the converse to the next section.

Lemma 2. *$[\![\,]\!]$ maps permutation equivalent proof terms to the same*[5] *tragr.*

Proof. We show for each law in Table 2 its left- and right-hand sides are mapped to the same tragr by $[\![\,]\!]$:

- For the monoid laws (h-left unit), (h-right unit) and (h-associativity) for horizontal composition, the former two follow from that the parallel composition of a tragr with $[\![\varepsilon]\!]$ on either side, amounts to first introducing and then immediately removing εs. Associativity holds since removing the εs and redirecting the respective input and output edges are local and independent actions.

[3] It is interesting to compute their respective tragrs and see that / how they differ.

[4] Our proof below for trace graphs follows that for trace relations [30, Lemma 8.6.14].

[5] Formally, the same up to graph isomorphism.

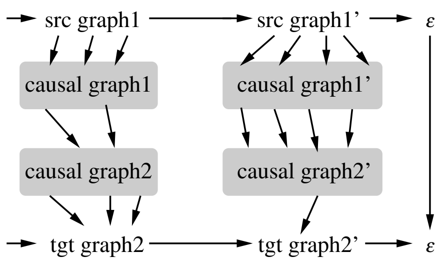


Figure 3: Tragr illustrating (exchange)

- For the monoid laws (v-left unit), (v-right unit) and (v-associativity) for vertical composition, the former two follow from that for any string s, $[\![s]\!]$ is a *ladder*, a tragr only comprising the straight line graphs of its source and target string, each the reverse of the other. For the sequential composition with a ladder on either side, elision amounts to the immediate removal of (the reverse of) the ladder. Associativity holds since elision is complete (confluent and terminating) and can be postponed until after connecting the respective input and output ports, which are local and independent actions.
- The (exchange) law holds by combining the reasoning in the previous two items; combining removal of εs with elision $\Rightarrow$ is complete and can be postponed until after redirecting the input and output edges, which are local and independent actions; see Figure 3. $\square$

We conclude this section with showing that any tragr can be read back into a multistep reduction, by means of a procedure we dub topological *multi*sorting, which is like topological sorting but selects in each stage *all* minimal elements, instead of just a *single* such, cf. [26].

Definition 7. *The* topological multisorting *function* TS *mapping a tragr from s to t to a multistep reduction having s as source and t as target, is defined by induction on size and cases on its causal graph.*

If the causal graph is empty, planarity of tragrs dictates the tragr is a ladder (as in the Proof of Lemma 2; cf. the bottom-right of Appendix A), so we have $s = t$ and may return the empty multistep s.

If the causal graph is non-empty, let its minimal layer *M comprise its minimal nodes w.r.t. the partial order induced by the dag. To construct the multistep Φ we juxtapose, starting from the input of the tragr, the labels (letters) of nodes in the dag for s not covered by nodes in M, interspersed with the labels (rule symbols) of those covering nodes in M. Let s' be the target of Φ, and consider the tragr obtained by replacing for every node labelled by some rule ρ in T the source dag of ρ by its target dag. By planarity it follows (cf. the top row of Appendix A) that the resulting tragr is from s' to t. Therefore, it suffices to vertically compose Φ with the TS-image of this tragr, which exists by the IH.*

In both case, we obtain a vertical composition of multisteps having s as source qnd t as target, giving rise to a multistep reduction after removing a trailing empty multistep.

Example 8. *Applying topological multisorting* TS *to the tragr on the right in Figure 2 gives rise to the* 6 *successive stages displayed in Appendix A.*

4 Greedy multistep reductions

We first give a standard algorithm for transforming a proof term into a permutation equivalent *greedy* one [7], and next show there is a bijection between such greedy multistep reductions and tragrs. From this we conclude, in a semantic way, that both constitute unique representatives of permutation equivalence classes of proof terms.

We give a novel description of greediness and the greedy algorithm of [7], based on the analogy with sorting and standardisation [14, 30] in the literature. In sorting, (adjacent) *inversions* are consecutive elements that are out-of-order, and in term rewriting, *anti-standard* pairs [14, 19] are consecutive steps in a reduction such that the latter is outside (to the left of) the former. Such pairs of out-of-order elements are of interest since they provide a *local* characterisation both of *being sorted*, i.e. the *absence* of such pairs, and of bringing the list / reduction *closer to being sorted*, by *permuting* the out-of-order pair. This makes both processes amenable to a rewriting approach, with bubblesort being an example for sorting and the extraction of the leftmost-contracted-redex being an example for standardisation [14, 12, 18, 30, 19, 5]. To make the greedy algorithm fit the mould, we define *loath* pairs as consecutive multisteps where some rule symbol in the 2nd is *not caused* by the 1st, so may be permuted up front, signalling non-greediness. This is phrased in terms of residuation; see Definition 4.

Definition 8. *A proof term is* greedy *if it is a multistep reduction without loath pairs, where a pair $\Phi\cdot\Psi$ of consecutive multisteps is* loath *if there is a step X co-initial with Φ such that $\Phi\subseteq X$ and having residual step $\psi := X/\Phi$ with $\psi\subseteq\Psi$.* Swapping X for $\Phi\cdot\Psi$ *then results in $X\cdot(\Psi/\psi)$. Exhaustive swapping followed by removing trailing empty multisteps yields a* greedy decomposition*.*

Example 9. *The multistep reduction γ' is greedy, but γ isn't as is clear from $AB\beta\cdot\overline{A\alpha\underline{AAB}\cdot AA\underline{\beta}}\cdot\beta AAB\cdot B\beta AAB\cdot\overline{\alpha\underline{AAB}AAB\cdot A\underline{\beta}AAB}$, where we have overlined its loath pairs, and underlined the rule symbols and their left-hand sides involved in swapping. The loath pair $A\alpha\underline{AAB}\cdot AA\underline{\beta}$ swaps into $A\alpha\underline{\beta}\cdot AA\underline{BAAB}$, and $\alpha\underline{AAB}AAB\cdot A\underline{\beta}AAB$ swaps into $\alpha\underline{\beta}AAB\cdot A\underline{BAAB}AAB$. As one may verify, exhaustive swapping yields $\gamma'\cdot ABAABAAB\cdot ABAABAAB$, hence a greedy decomposition of γ is γ'. Intuitively, this is as desired since γ' exhibits maximal concurrency while performing the same tasks performed in γ.*

By standard residual theory [30, Chapter 8], swapping yields a pair of consecutive multisteps permutation equivalent to the original pair, as in the example. Moreover, the size (qua number of rule symbols) of the 2nd multistep decreases per construction, so swapping decreases the *Sekar–Ramakrishnan measure* [30, Definition 8.5.17], measuring a multistep reduction by the lexicographic product of the sizes of the multisteps in it from tail to head. Since if necessary we may first transform a proof term into a permutation equivalent (single step hence multistep) *reduction* by the Logicality Lemma 1, we have:

Lemma 3. *A proof term can be transformed into a permutation equivalent greedy multistep reduction.*

Remark 9. *To give an idea how residual theory [30, Table 8.5] may be employed to show swapping preserves permutation equivalence, first note that $\Phi\subseteq X$ entails Φ/X is an empty multistep. Therefore, by commutativity of join $X\equiv X\cdot(\Phi/X)\equiv\Phi\cdot(X/\Phi)$. Similarly, $X/\Phi=\psi\subseteq\Psi$ entails $\Psi\equiv(X/\Phi)\cdot(\Psi/(X/\Phi))$. By combining both $\Phi\cdot\Psi\equiv\Phi\cdot(X/\Phi)\cdot(\Psi/(X/\Phi))\equiv X\cdot(\Psi/(X/\Phi))=X\cdot(\Psi/\psi)$.*

Remark 10. *An efficient procedure for searching for loath pairs can be based on the observation that due to linearity of string rewrite systems, an occurrence of either a source or target of a rule can be identified with a* pattern *in the sense of [30, Definition 8.6.21], i.e. with a* convex *set of positions in the tree of the string having vertices as boundary. Following the main idea of [23], to see whether $\Phi\cdot\Psi$ is loath, it therefore suffices to check whether each pattern of a* source *of a rule occurring in Ψ has overlap with some* target *of a rule occurring in Φ. Since a pattern in a string simply is an* interval*, characterised by the two vertices constituting its boundary, a single top–down pass through both string-trees checking disjointness of intervals via their boundaries, suffices. If for some pattern there is no overlap, we obtain a loath pair by setting X to Φ in which the pattern was replaced by the rule.*

For example, using underlining to indicate occurrences of patterns, that $\alpha AABAAB, A\beta AAB$ *in* γ *is a loath pair follows from that the pattern* $\{\mathring{2},\overline{3},\mathring{3},\overline{4},\mathring{4}\}$ *in* $A\underline{AAB}AAB$ *corresponding to the source* AAB *of the rule* β*, does not have overlap with the pattern* $\{\mathring{1}\}$ *in* $\underline{A}AABAAB$ *corresponding to the target* A *of the rule* α*. This in turn follows from that the corresponding intervals* $[\mathring{2},\mathring{4}]$ *and* $[\mathring{1}]$ *are disjoint since* $\mathring{1}$ *is smaller than* $\mathring{2}$*. By disjointness / non-overlap, replacing in* $\alpha\underline{AAB}AAB$ *the* β*-pattern* AAB *by the rule* β *yields the multistep* $\alpha\beta AAB$*, as desired.*

Theorem 1. *There is a(n effective) bijection between greedy multistep reductions and tragrs.*

Proof. We show that TS and $[\![\,]\!]$ are maps from tragrs to greedy multistep reductions and from greedy multistep reductions, respectively, and are inverse to each other.

- We show the composition of $[\![\,]\!]$ with TS is the identity by induction on the length of a greedy multistep reduction. We employ the no(ta)tions of Definition 7, in particular we employ M to denote the layer of minimal elements of (the causal graph of) a tragr.

  For the empty and single-multistep reductions this is trivial. Otherwise, the reduction has shape $\Phi \cdot \gamma$. By definition $[\![\Phi \cdot \gamma]\!]$ is the serial composition of $[\![\Phi]\!]$ and $[\![\gamma]\!]$ and we claim that by greediness the steps in the minimal layer $[\![\Phi \cdot \gamma]\!]$ of the tragr are those of $[\![\Phi]\!]$, i.e. $\mathsf{M}([\![\Phi \cdot \gamma]\!]) = \mathsf{M}([\![\Phi]\!])$. Then, Φ is the result of the first stage of TS and $\mathsf{TS}([\![\Phi \cdot \gamma]\!]) = \Phi \cdot \mathsf{TS}([\![\gamma]\!]) = \Phi \cdot \gamma$ by the IH for γ.

  It remains to prove the claim that $\mathsf{M}([\![\Phi \cdot \gamma]\!]) = \mathsf{M}([\![\Phi]\!])$ for a greedy multistep reduction of shape $\Phi \cdot \gamma$, so with γ non-empty. Since $\mathsf{M}([\![\Phi \cdot \gamma]\!]) \supseteq \mathsf{M}([\![\Phi]\!])$ trivially holds, for arbitrary multistep reductions, suppose for a proof by contradiction that $\mathsf{M}([\![\Phi \cdot \gamma]\!]) \subseteq \mathsf{M}([\![\Phi]\!])$ does not hold, for $\Phi \cdot \gamma$ of minimal length. Then there must be some node in $\mathsf{M}([\![\gamma]\!])$ in $\mathsf{M}([\![\Phi \cdot \gamma]\!])$, per construction of $[\![\Phi \cdot \gamma]\!]$ as the serial composition of $[\![\Phi]\!]$ and $[\![\gamma]\!]$. By minimality this node must in fact be in $\mathsf{M}([\![\Psi]\!])$ for Ψ the first multistep of γ, with the node corresponding to, say, step $\psi \subseteq \Psi$. But then $\Phi \cdot \Psi$ would be a loath pair, as it allows swapping the join of Φ with ψ.[6] This contradicts the assumed greediness of $\Phi \cdot \gamma$.

- For the converse direction, we first show that when computing the TS-image of a tragr, consecutive stages yield multisteps that are not loath pairs, by induction on the number of stages. There is only something to show when there is more than one stage. So suppose TS yields a composition $\Phi \cdot \gamma$ with Φ obtained from the minimal layer of rule nodes M of the tragr, and $\gamma\cdot$ from its remaining nodes / causal graph R, non-empty by assumption. By the IH γ is greedy, and non-empty so has some first multistep, say Ψ, constructed from the minimal layer, say N, of R. Per definition of TS each of the nodes in N is reachable from some node in M. Since there are no edges between the nodes in a single layers, this entails that for each of the nodes in N there is an edge to it from some node in M. As a consequence, cf. again Remark 10, the corresponding pair $\Phi \cdot \Psi$ of consecutive multisteps is greedy / not loath.

  From this it easily follows that every tragr is mapped to itself, by induction on the number of stages, by showing that for Φ the multistep of a stage, the tragr $[\![\Phi]\!]$ yields the stage again. □

We can now establish our main result, that one may compute a greedy multistep reduction, unique modulo permutation equivalence, for any proof term by first evaluating into its tragr / causal graph (using $[\![\,]\!]$), followed by the topological multisort (using TS) yielding the greedy multistep reduction.

[6]More precisely, the join of Φ with the origin of ψ along the converse of Φ, which is a step acting on an interval in the dag of the source string of Φ, as observed in Remark 10. Note our reasoning would fail if rules were allowed to have empty left- or right-hand sides: If ψ were due to a rule with an empty left-hand side, or if Φ were to contain a rule with an empty right-hand side, then X might not be swappable.

Theorem 2. *For every proof term γ, there exists a unique greedy multistep reduction γ' such that $\gamma \equiv \gamma'$.*

Proof. Lemma 3 shows existence. To show uniqueness, consider greedy multistep reductions γ' and γ'' both permutation equivalent to γ. By Lemma 2, $[\![\gamma']\!]$ and $[\![\gamma'']\!]$ are the same tragr. Therefore, $\gamma' = \mathsf{TS}([\![\gamma']\!]) = \mathsf{TS}([\![\gamma'']\!]) = \gamma''$ by TS being inverse to $[\![\,]\!]$ on greedy multistep reductions by Theorem 1.. $\square$

Remark 11.
- *The proof only employs one direction (the first half of the proof) of Theorem 1.*
- *As a consequence, using that the greedy multistep reductions* are *the normal forms w.r.t. swapping, we have that swapping is confluent on multistep reductions. This could alternatively be established via Newman's Lemma, using that swapping is terminating and showing locally confluence.*

Example 10. *The greedy multistep reduction γ' is the unique representative of the permutation equivalence class of γ. Both are mapped to the tragr on the right in Figure 2 by the proof term algebra $[\![\,]\!]$, and topological multisorting*

5 Conclusions

We have given further support to that, as stated in the introduction, notions of causal equivalence are omnipresent. Here we have shown that Lévy's notion of permutation equivalence [17] as known from *term rewriting* [30] corresponds, after specialising it to *string rewriting*, to the notion of causal equivalence as employed by Wolfram in his *physics* project [32, 33].

This we achieved by introducing tragrs, refining Wolfram's notion of causal graph, as representatives of permutation equivalence classes of reductions. Representing reductions as term themselves, so-called proof terms [20], allowed us to specify the representation map, from reductions to tragrs, effectively by means of a (proof term) algebra that models permutation equivalence. To show representatives unique, we gave a map back from tragrs to so-called greedy multistep reductions as known from Dehornoy's work in *algebra* [7], using a topological multisorting procedure, showing both maps to be inverse to each other.

We think that giving different perspectives on the *same* notion, as we did for causal equivalence here but also before in [30, Chapter 8], is important. Hence we also think it unfortunate to (re)invent wheels without noticing wheels to be the *same*. That seems to be the situation for causal equivalence though; in the literature as given in the introduction cross-references beyond the borders of the specific field (to name a few: rewriting, algebra, physics, category theory, proof theory, concurrency theory) of a paper are few and far between. We hope our short paper can contribute to creating at least some awareness of that unfortunate situation for causal equivalence, and the need to overcome it.[7]

The concepts and techniques developed and employed here are simple.[8] For instance, topological multisorting could be easily presented in undergraduate Discrete Mathematics or Data Structures and Algorithms courses. We view this as a strength rather than as a weakness. Only *because* the results are simple in the linear case of string rewriting do we entertain hope to extend them to non-linear cases, e.g. to our main field of study term rewriting.

All results here are effective / constructive, but we didn't study the complexity of them. However, we do hope the concrete representations of permutation equivalence classes by means of tragrs (certain graphs) and greedy multistep reduction (certain terms) could be useful for such, cf. [7].

[7]Several people we spoke to expressed views such as: "that paper is not relevant since it is on concurrency theory not on rewriting", and then didn't think their view to be problematic.

[8]This could also be the reason for the observed disjointedness of the literature on causal equivalence: simplicity allows to reinvent in an *ad hoc* way.

Log and Acknowledgments This short paper was provoked by the remark made to me in 2020[9] by Jan Willem Klop that Wolfram's causal graphs should characterise permutation equivalence. Being aware of that, my short reply then was to refer him to trace relations [30] and drawing Figure 2. Realising a single picture was too cryptic, that was expanded[10] into a 10-page submission to the workshop Termgraph 2022.

Compared to the workshop version, the present short paper includes (more detailed and simplified) proofs and more illustrations. To make space for them we have now exclusively focussed on the oudenadic embedding, omitting results on the monadic embedding (cf. Remark 5), in particular omitting the result that permutation equivalence as defined in Table 2 for the oudenadic embedding corresponds (via transformations back and forth) to permutation equivalence as defined in [30, Table 8.2] for the monadic embedding.

We thank Jan Willem Klop for suggesting the topic is of interest, and Nao Hirokawa and the reviewers and participants of Termgraph 2022 in Haifa for feedback.

Remark 12. *The URLs in the references were verified to work on the 2nd of November 2022.*

References

[1] F. Baader & T. Nipkow (1998): *Term Rewriting and All That*. Cambridge University Press.

[2] P. Barenbaum & E. Bonelli (To appear): *Reductions in Higher-Order Rewriting and Their Equivalence*. In: *CSL 2023*.

[3] A. Bawden (1986): *Connection Graphs*. In: *LFP 1986*, ACM, pp. 258–265, doi:10.1145/319838.319868.

[4] G. Boudol (1985): *Computational semantics of term rewriting systems*. In M. Nivat & J.C. Reynolds, editors: *Algebraic Methods in Semantics*, Cambridge University Press, pp. 169–236.

[5] H.J.S. Bruggink (2008): *Equivalence of Reductions in Higher-Order Rewriting*. Ph.D. thesis, Utrecht University. Available at `http://dspace.library.uu.nl/handle/1874/27575`.

[6] A. Church & J.B. Rosser (1936): *Some properties of conversion*. *Transactions of the American Mathematical Society* 39, pp. 472–482, doi:10.1090/S0002-9947-1936-1501858-0.

[7] P. Dehornoy & alii (2015): *Foundations of Garside Theory*. European Mathematical Society, doi:10.4171/139.

[8] A. Guglielmi, T. Gundersen & M. Parigot (2010): *A Proof Calculus Which Reduces Syntactic Bureaucracy*. In: *RTA 2010*, *LIPIcs* 6, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 135–150, doi:10.4230/LIPIcs.RTA.2010.135.

[9] T. Hirschowitz (2013): *Cartesian closed 2-categories and permutation equivalence in higher-order rewriting*. *LMCS* 9(3), doi:10.2168/LMCS-9(3:10)2013.

[10] G. Huet & J.-J. Lévy (1991): *Computations in Orthogonal Rewriting Systems, Part I + II*. In J.L. Lassez & G.D. Plotkin, editors: *Computational Logic – Essays in Honor of Alan Robinson*, MIT Press, Cambridge MA, pp. 395–443. Update of: Call-by-need computations in non-ambiguous linear term rewriting systems, 1979.

[11] A. Joyal & R. Street (1991): *The geometry of tensor calculus, I*. *Advances in Mathematics* 88(1), pp. 55–112, doi:https://doi.org/10.1016/0001-8708(91)90003-P.

[12] Z. Khasidashvili & J.R.W. Glauert (1996): *Discrete Normalization and Standardization in Deterministic Residual Structures*. In: *ALP'96*, *LNCS* 1139, Springer, pp. 135–149, doi:10.1007/3-540-61735-3_9.

[13] Z. Khasidashvili & J.R.W. Glauert (2002): *Relating conflict-free stable transition and event models via redex families*. *TCS* 286(1), pp. 65–95, doi:10.1016/S0304-3975(01)00235-3.

[9]While employed as UniversitätsassistentIn in the Computational Logic group at the University of Innsbruck.

[10]While employed as Research Associate in the Mathematical Foundations of computation group at the University of Bath.

[14] J.W. Klop (1980): *Combinatory Reduction Systems*. Ph.D. thesis, Rijksuniversiteit Utrecht.

[15] Y. Lafont (1990): *Interaction Nets*. In: 17*th POPL*, ACM Press, pp. 95–108, doi:10.1145/96709.96718.

[16] C. Laneve (1994): *Distributive Evaluations of lambda-calculus*. *Fundam. Informaticae* 20(4), pp. 333–352, doi:10.3233/FI-1994-2043. Available at `https://doi.org/10.3233/FI-1994-2043`.

[17] J.-J. Lévy (1978): *Réductions correctes et optimales dans le λ-calcul*. Thèse de doctorat d'état, Université Paris VII. Available at `http://pauillac.inria.fr/~levy/pubs/78phd.pdf`.

[18] P.-A. Melliès (1996): *Description Abstraite des Systèmes de Réécriture*. Thèse de doctorat, Université Paris VII. Available at `http://www.irif.fr/~mellies/phd-mellies.pdf`.

[19] P.-A. Melliès (2005): *Axiomatic Rewriting Theory I: A Diagrammatic Standardization Theorem*. In: *Essays Dedicated to Jan Willem Klop*, *LNCS* 3838, Springer, pp. 554–638, doi:10.1007/11601548_23.

[20] J. Meseguer (1992): *Conditional rewriting logic as a unified model of concurrency*. *Theoretical Computer Science* 96, pp. 73–155, doi:10.1016/0304-3975(92)90182-F.

[21] M.H.A. Newman (1942): *On theories with a combinatorial definition of "equivalence"*. *Annals of Mathematics* 43, pp. 223–243, doi:10.2307/2269299.

[22] V. van Oostrom (2004): *Sub-Birkhoff*. In: *FLOPS 2004*, *LNCS* 2998, Springer, pp. 180–195, doi:10.1007/978-3-540-24754-8_14.

[23] V. van Oostrom (2020): *Some symmetries of commutation diamonds*. In: *IWC 2020*, pp. 1–7. Available at `http://iwc2020.cic.unb.br/iwc2020_proceedings.pdf`.

[24] V. van Oostrom, K.J. van de Looij & M. Zwitserlood (2004): *Lambdascope*. In: *ALPS 2004*, p. 9. Available at `http://cl-informatik.uibk.ac.at/users/vincent/research/publication/pdf/lambdascope.pdf`.

[25] G.D. Plotkin (1980s): *Notes on Church–Rosser categories*. Unpublished manuscript obtained via Jan Willem Klop in 2022.

[26] G.D. Plotkin & V.R. Pratt (1996): *Teams can see pomsets*. In D.A. Peled, V.R. Pratt & G.J. Holzmann, editors: *Partial Order Methods in Verification, Proceedings of a DIMACS Workshop, Princeton, New Jersey, USA, July 24-26, 1996*, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 29, DIMACS/AMS, pp. 117–128, doi:10.1090/dimacs/029/07.

[27] T. Rowland & E.W. Weisstein: *Causal Network*. Available at `https://mathworld.wolfram.com/CausalNetwork.html`.

[28] E.W. Stark (1989): *Concurrent Transition Systems*. *TCS* 64, pp. 221–269, doi:10.1016/0304-3975(89)90050-9.

[29] C.A. Stewart (2002): *Reducibility between Classes of Port Graph Grammar*. *J. Comput. Syst. Sci.* 65(2), pp. 169–223, doi:10.1006/jcss.2002.1814.

[30] Terese (2003): *Term Rewriting Systems*. Cambridge University Press.

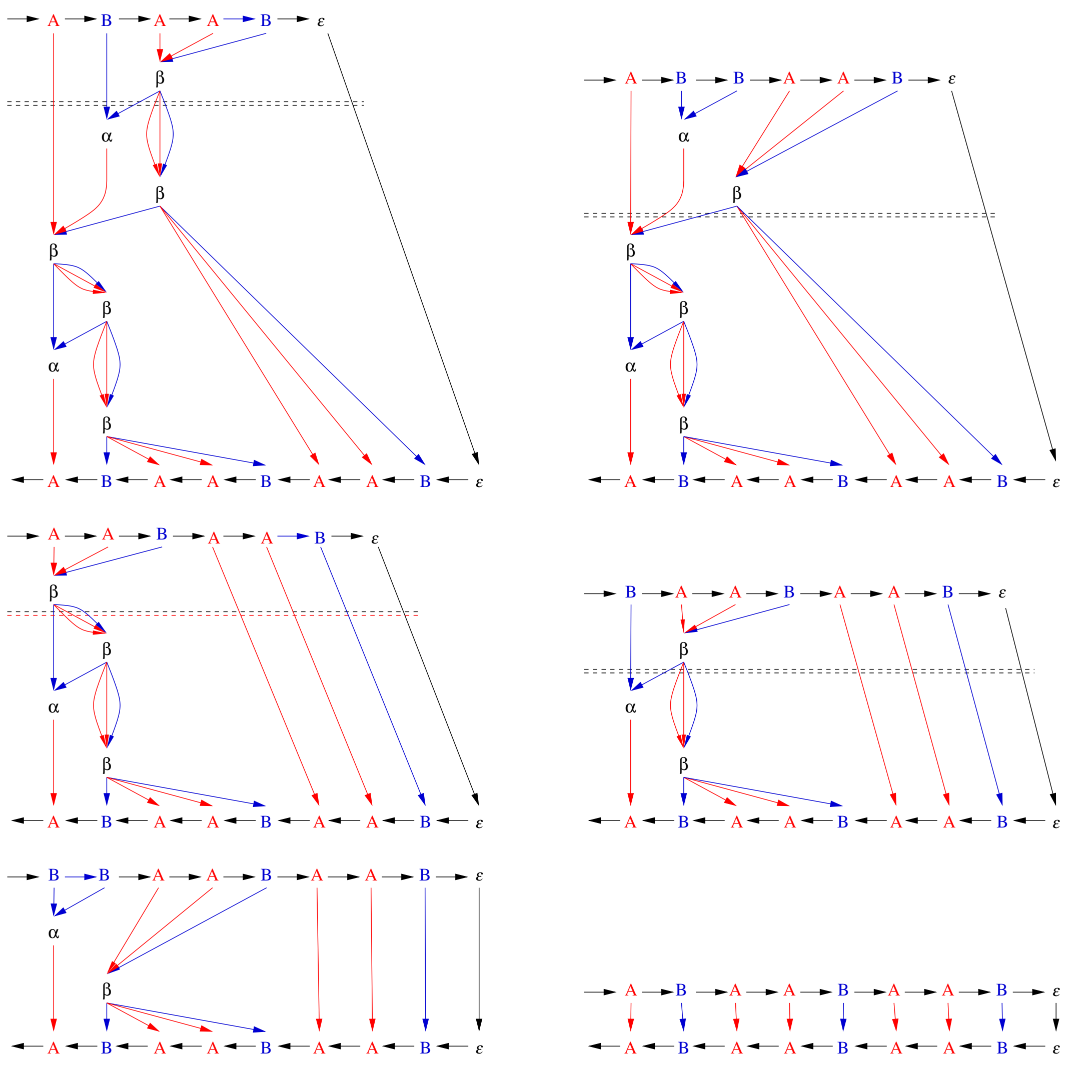
[31] G. Winskel (1989): *An introduction to event structures*. In: *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, *LNCS* 354, Springer, pp. 364–397, doi:10.1007/BFb0013026.

[32] S. Wolfram (2002): *A New Kind of Science*. Available at `https://www.wolframscience.com/nks/`.

[33] S. Wolfram (2020): *A Class of Models with the Potential to Represent Fundamental Physics*. Available at `https://www.wolframphysics.org/technical-introduction/`.

A Topologically multisorting the tragr of Figure 2 (right)

Reading back the tragr in Figure 2 (right) by topological multisorting gives rise to the following 6 stages. Vertically composing the corresponding 5 multisteps (not taking the last empty multistep into account) yields the multistep reduction $AB\beta \cdot A\alpha\beta \cdot \beta AAB \cdot B\beta AAB \cdot \alpha\beta AAB$. That is, we have read back γ'!


A B A A B ε
β
α
β
A B A A B A A B ε
A B B A A B ε
A A B A A B ε
B A A B A A B ε
B B A A B A A B ε
A B A A B A A B ε