On Causal Equivalence by Tracing 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

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. In [35, Chapter 8] we presented five conceptually distinct ways to mathematically model *causal* equivalence [35, Section 8.1.3] of computations in term rewriting, based on *permutation*, *labelling*, *standardisation*, *extraction* and *projection*, respectively, and showed them to coincide.

Though coincidence of the above five perspectives gave us confidence in having captured causal equivalence within term rewriting, at the time we failed to relate them to the further perspective put forward there based on *tracing* [35, Section 8.6]. The problem[1] to do so resides in term rewrite rules that are *non-linear*. For example, consider a reduction $f(a) \to g(a,a) \to g(b,a) \to g(b,c)$ in the term rewrite system having rules $a \to b$, $a \to c$ and $f(x) \to g(x,x)$, with the last rule non-linear (it *copies* x). The single occurrence of the symbol a in $f(a)$ *traces*[2] to both occurrences of a in $g(a,a)$, hence *causes* both subsequent steps. However, the fact that a was *copied* by the first step is *not* captured by tracing; f traces to neither copy of a, though the first step rewriting f is the *cause* of having the two further steps rewriting the copies of a, in the first place. In this paper we show that only non-linearity is problematic for tracing. More precisely, we show that for string rewriting, which is inherently *linear*, causal equivalence does have a simple characterisation based on tracing, namely by *tragrs*.[3]

In Section 2 we adapt the theory of *permutation* equivalence of finite computations as developed in [35, Chapter 8] for term rewriting, to the case of string rewriting at hand. To represent, interpret, and prove properties about, finite computations by algebraic and inductive means, we adapt *proof* terms [35, Chapter 8] to represent the finite reductions of a string rewrite system. In Section 3 we introduce tragrs as a formalisation of Wolfram's notion of *causal graph* [39]. We relate proof terms to tragrs by on the one hand giving an algebra $[\![\,]\!]$ interpreting proof terms as tragrs preserving permutation equivalence. and on the other hand presenting a topological multi-sorting algorithm TS mapping tragrs back to proof

*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] Cf. the paragraph on top of page 415 in [35, Remark 8.6.74].

[2] For the *trace* relations presented in [35, Section 8.6.1]; see Definitions 8.6.7, 8.6.17, Lemma 8.6.14, and Proposition 8.6.18.

[3] Tragr is short for *tra*ce *gr*aph and pronounced as *tracker*, with the idea that a tragr *tracks* what happens to symbols.

C. Grabmayer (Ed.): Post-Proceedings of TERMGRAPH 2022
EPTCS 377, 2023, pp. 27–43, doi:10.4204/EPTCS.377.2

terms. In Section 4 we show that any proof term may be transformed, by repeatedly swapping *loath* pairs,[4] into a permutation equivalent *greedy* [7] multistep reduction, and that the latter are in bijective correspondence to tragrs via the maps $[\![\,]\!]$ and TS. This then allows us to wrap up and conclude that tragrs (and greedy multistep reductions) serve as unique representatives of permutation equivalence classes.

Remark 1. *This short paper was provoked by a remark made to me*[5] *in 2020 by Jan Willem Klop, that Wolfram's causal graphs [39] should characterise permutation equivalence, for string rewriting. Having followed Wolfram's* physics *project myself and having observed that its developments frequently ran parallel to those in [35, Chapter 8], in particular that there was a close connection between causal graphs and the trace relations in [35, Section 8.6.1] discussed above, allowed me to reply immediately, confirming Klop's intuition, referring him to [35, Chapter 8] and drawing Figure 2. At the time I was reluctant to develop that further, as the idea was simple and did not solve the problem for the non-linear case left open in [35]. Later, in 2022,*[6] *I realised that a single picture was too cryptic, and that only* because *the results are simple here do we entertain hope to extend them to the complex non-linear case.*

2 Proof terms for string rewriting

We adapt the theory of permutation equivalence from term rewriting [35, Chapter 8] to string rewriting, 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 string rewrite rules to be the empty string. This restriction makes sense from the perspective of causality [37] as it entails all steps being *ex materia* (forbidding *ex nihilo* steps) and having *bounded* causes; cf. the 4th item of Remark 9. 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 mentioned in the introduction, and these perspectives can in turn be proven equivalent in a *simple* way due to the absence of replication. As in [35, Chapter 8], to state and prove results *proof* terms are our tool of choice for representing the reductions of a string rewrite system.

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.

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 $(s \cdot_h t) \cdot_h u = s \cdot_h (t \cdot_h u)$.*

Remark 2. *Our terminology* oudenadic *is an attempt to highlight that the representation employed here associates* nullary *function symbols to letters, and to contrast it with the usual* monadic *representation, which associates a* unary *function symbol to each letter; cf. [35, Section 3.4.4] for an account of both.*

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 $a_1 \ldots a_n$ for some $n \geq 1$.

[4] A loath pair intuitively is a pair of consecutive steps that are causally independent, so that the second step could have been in parallel to the first, but it was too *loath* to do so.

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

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

We refer to $\cdot_h$ as *horizontal* composition to distinguish it from *vertical* composition $\cdot_v$ below (see Definition 2), based on that we adhere to a convention of drawing strings horizontally and reductions vertically in figures. That meshes well with thinking of strings as being extended in space (1-dimensional, horizontally) and of reductions as extended in time (vertically); cf. Figure 1. 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*: if $a_1 \dots a_n = b_1 \dots b_m$ for $a_i, b_j \in \Sigma$, then $n = m$ and $a_k = b_k$ for all $1 \leq k \leq n$, cf. [36].

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.,* $(AB)(AA)B$*).*

The alphabet $T := \{AB, B, A\}$ *does not have unique reading, e.g.* ABB *can be viewed as being composed of the two letters* AB *and* B*, and alternatively of the three letters* A*,* B *and* 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 $\equiv_M$. We use $a, b, \dots$ as variables for letters, $A, B, \dots$ as concrete letters, and A, B for the concrete letters of our running example, as in Example 1.

We consider term rewrite systems in the sense of [35, 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 [21], as *proof* terms [35], terms over a signature comprising the letters, the rules, and a binary composition symbol $\cdot_v$ representing the transitivity inference rule of rewrite logic [21]. 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, in Table 1, 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)	$\varepsilon :$	$\varepsilon \geqslant \varepsilon$	
(letter)	$a :$	$a \geqslant a$	for each letter a
(rule)	$\rho :$	$\ell \geqslant r$	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)	$\gamma \cdot \delta :$	$s \geqslant u$	if $\gamma : s \geqslant t$, and $\delta : t \geqslant u$

Table 1: Proof terms for string rewriting

We abbreviate vertical composition $\cdot_v$ to $\cdot$, assume it is right-associative, and that it binds weaker than horizontal composition $\cdot_h$ / juxtaposition.

Remark 3. *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 can 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.

(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 inducing permutation equivalence

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* reduction *(a* multistep reduction*) either is an empty multistep or a vertical composition, associated to the right, of steps (resp. nonempty multisteps).* Permutation *equivalence $\equiv$ between proof terms is induced by the laws in Table 2, where the sides of the laws are restricted to proof terms, i.e. sources and targets of the proof terms γ,δ,ζ,η and the oudenadic terms s,t 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. *The requirements on the sources and targets in the laws of Table 2 boil down to working in a* typed *algebraic structure [29]. For instance, in that setting a* category *is a typed* monoid*, allowing composition of morphisms only if their sources and targets match. In Table 2 the requirements are most prominent in the (exchange) law. For example, for rules $\alpha:A\to A'C,\alpha':A'\to A'',\beta:B\to B',\beta':CB'\to B''$, though we do have $\alpha\beta\cdot\alpha'\beta':AB\geqslant A''B''$, the expression $(\alpha\cdot\alpha')(\beta\cdot\beta')$ is not even a proof term since, e.g., the target $A'C$ of α does not match the source A' of α'.*

Remark 5. *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 [2, 35]. 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 $\multimap$ of [35, Chapter 8] for multisteps:

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

Logicality, cf. [23], 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 we 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, $\zeta\langle u\rangle$ denotes the reduction obtained by suffixing each step of ζ by u, and symmetrically for $\langle u\rangle\zeta$. One easily verifies $\gamma' : s_1s_2 \geqslant t_1t_2$, and also that $\gamma \equiv \gamma'$ by using (exchange) and vertical units 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 transforms 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 6. *Logicality is the raison d'être for the field of rewriting [2, 35], allowing to reduce reachability to reducibility. Cf. [21, 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 in 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 2nd and 3rd 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.

(too large) (Multi)steps not only represent what changes (via the rules in it) but also what *does not change* (via the letters in it); cf. the frame problem [31]. As a consequence, in general proof terms predominantly consist of letters; this holds true in particular both for γ and γ'. *Causal graphs* [38]

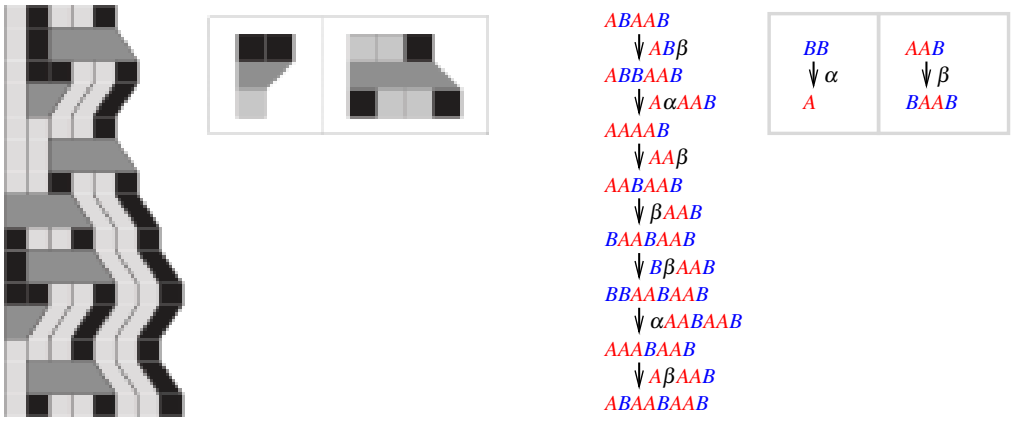


Figure 1: Reduction $\gamma\colon ABAAB \geqslant ABAABAAB$ (right) and its evolution (left)

(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 [35] 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 causal graphs [38]. The algebra is shown to model permutation equivalence in that it maps permutation equivalent proof terms to the same tragr. We give a procedure we dub topological multi-sorting, 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* [38], 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 [30]; based on [38, 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 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).*

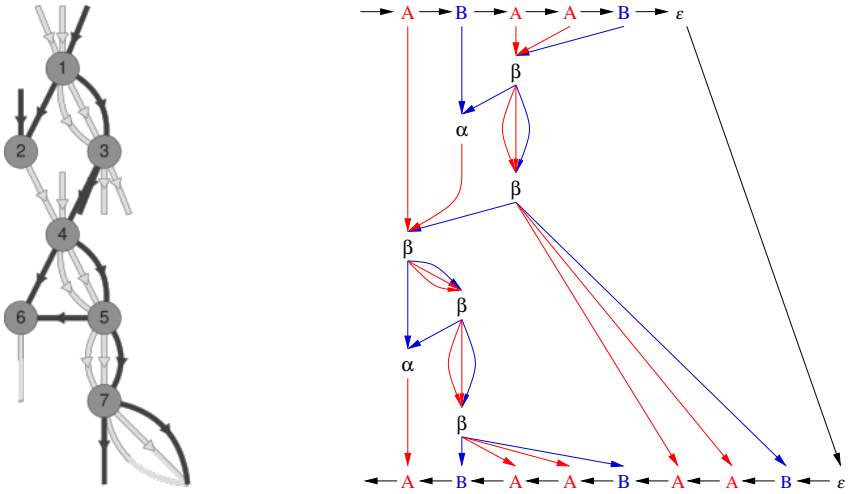
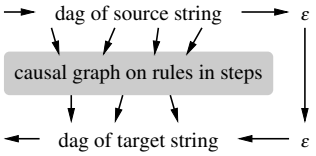


Figure 2: Causal graph (left) and tragr from $ABAAB$ to $ABAABAAB$ (right)

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*, short for *trace graph*, illustrated for γ / γ' in Figure 2.

Remark 7. *The book [38] 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 [34] (see also [3, 16, 34, 25]). comprising the following three parts, as visualised in:*



- *the dag of source string s having for every occurrence of a letter b in s a node labelled b, having (in clockwise order) an input port of type*[7] $*$*, an output port of type* $*$*, and an output port of type b. 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*.*
- *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.*

[7] Types serve here only to enable indicating / visualising connections between ports and edges conveniently (cf. Example 6).

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. (See Remark 10 for more on the planarity requirement.)

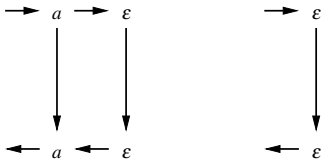
We indicate *the* input / output ports of a tragr by dangling edges, and refer to the dags of the source and target strings combined as its *interface*.

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

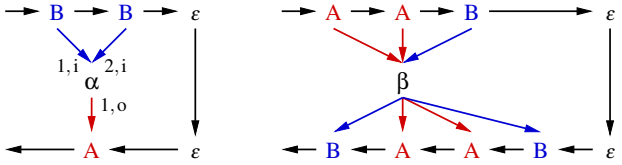
Remark 8. *Tragrs are not (too large) in the sense discussed above; letters only feature in the interface but not in the causal graph of a tragr; cf. the text below [35, Def. 8.6.17].*

Definition 6. *For a string rewrite system (Σ, P) the proof term algebra $[\![\,]\!]$ on tragrs, interpreting each proof term $\gamma\colon s \geqslant t$ as a tragr $[\![\gamma]\!]$ from s to t, is given by:*

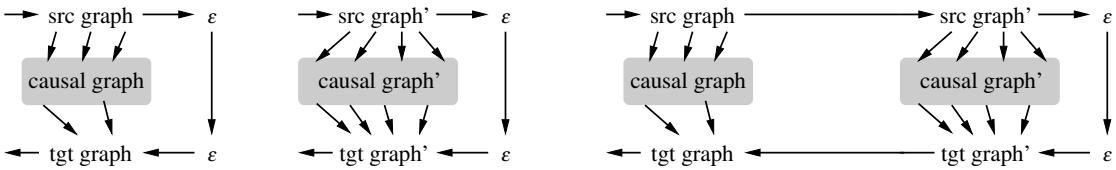
(letter and empty) $[\![a]\!]$ and $[\![\varepsilon]\!]$ are the tragrs:



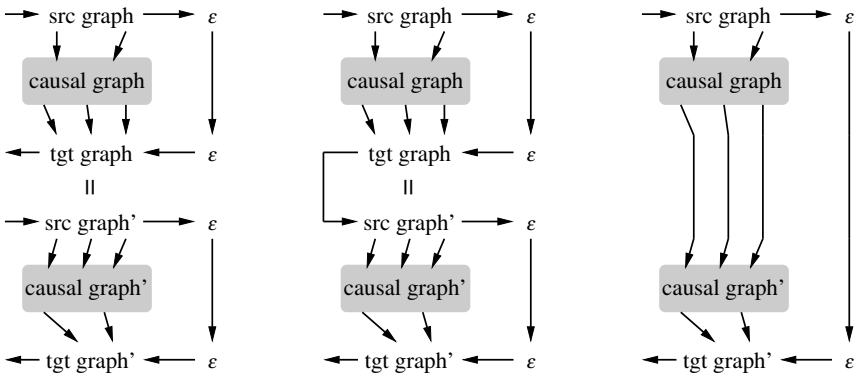
(rule) $[\![\rho]\!]$ is a tragr having the straight line dags for its source and target as interface, comprising a single rule node connected to the interface in an orderly way, illustrated for rules α and β by:



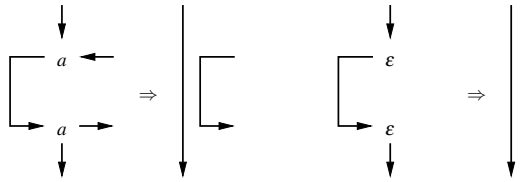
(juxtaposition) $[\![\gamma\delta]\!]$ is obtained from $[\![\gamma]\!]$ and $[\![\delta]\!]$ by removing the εs from the former, and redirecting the input and output of the latter accordingly:



(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 $[\![\gamma]\!]$ indeed 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 9.
- *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 [16].*
- *$[\![\gamma]\!]$ 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 [35, 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.*[8]
- *If we were to allow right-hand sides to be empty as in $\alpha\colon A \to \varepsilon$, then the number of occurrences of α that may cause the left-hand side of another rule may be unbounded as illustrated by the multisteps of shape $B\alpha^n B\colon B\underline{A^n}B \geqslant BB$ for rule $\beta\colon BB \to \ldots$ and any $n \in \mathbb{N}$, despite that none of the As trace to the rule β (its left-hand side BB).*[9] *Swapping left- and right-hand sides in this example illustrates the problem with allowing left-hand sides to be empty.*
- *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,[10] see Example 7, but defer showing the converse to the next section (see Theorems 1 and 2).

Lemma 2. *$[\![\,]\!]$ maps permutation equivalent proof terms to the same*[11] *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.
- 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

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

[9] Cf. [35, Definition 8.6.64] for a hack to overcome (by reifying 'emptyiness') the problem caused by such *collapsing* rules.

[10] Stated differently, we show the proof term algebra $[\![\,]\!]$ is a *model* (in an appropriate typed sense of [35, Definition 2.5.1(v)]) of permutation equivalence given in Table 2. The proof for trace graphs follows that for trace relations [35, Lemma 8.6.14].

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

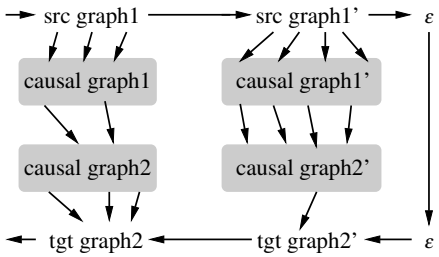


Figure 3: Tragr illustrating (exchange)

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; [12] see Figure 3. □

We conclude this section with showing that any tragr can be read back into a multistep reduction, by means of a procedure that we dub topological *multi*sorting, which is analogous to topological sorting but selects in each stage *all* minimal elements, instead of just a *single* one, cf. [28].

Definition 7. *The* topological multi-sorting *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 (taking the reflexive–transitive closure of) 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 cases, we obtain a vertical composition of multisteps having s as source and t as target, giving rise to a multistep reduction after removing a trailing empty multistep.

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

Remark 10. *Note how the planarity requirement on tragrs of Definition 5 was used in Definition 7 to guarantee that all tragrs having an empty causal graph read back as the empty multistep on their source (and target). In particular, planarity disallows 'rewirings' having crossing edges.*

Lemma 3. $[\![]\!]$ *after* TS *is the identity on tragrs.*

[12] For the reasoning to apply it is essential that *both* sides of (exchange) law are proof terms, i.e. that sources and targets of the constituting proof terms should match appropriately so that both sides indeed yield tragrs; cf. Remark 4.

Proof. By induction on size and cases on the causal graph of a tragr from s to t.

If the causal graph is empty then by the observation in Definition 7 and Remark 10, the tragr is a ladder, $s = t$, and it is mapped to the empty multistep s. Then $[\![s]\!]$ is *the* ladder from s to s again, as follows by induction on the length of s and Definition 6, using (letter) / (empty) in the base case and (juxtaposition) in the induction step.

If the causal graph is not empty, then per the construction in Definition 7, it is obtained by (transitivity) from the tragr from s to s' of its minimal layer M, and the tragr from s' to t of its remaining nodes / causal graph R. We conclude by that the former is obtained from $[\![\Phi]\!]$ for $\Phi : s \geqslant s'$ the multistep constructed from M in Definition 6, and by the induction hypothesis for the latter. To see the former, one proceeds as for the empty causal graph additionally using (rule) in the base case. □

Remark 11. *In fact,* any *way to transform a tragr into a proof term by repeatedly* decomposing *the tragr by means of the* inverses *of (juxtaposition) and (transitivity), mirrored by* composing *the corresponding proof terms by means of horizontal and vertical composition respectively, and transforming the base cases (letter), (empty) and (rule) in the natural way, will preserve the result (Lemma 3). The particular such transformation* TS *was chosen here because it does not just yield* any *proof term but a* greedy multistep reduction*, which will be essential for the unique representation purposes of the next section.*

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 [15, 35] in the literature. In sorting, (adjacent) *inversions* are consecutive elements that are out-of-order, and in term rewriting, *anti-standard* pairs [15, 20] 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 [15, 13, 19, 35, 20, 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* $A\alpha\underline{\beta} \cdot AA\underline{B}AAB$*, and* $\alpha\underline{AAB}AAB \cdot A\underline{\beta}AAB$ *swaps into* $\alpha\underline{\beta}AAB \cdot A\underline{B}AA\underline{B}AAB$*. 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 [35, Chapter 8], swapping yields a pair of consecutive multisteps permutation equivalent to the original pair, as in the example. Moreover, the size (number of rule symbols) of the 2nd multistep decreases per construction, so swapping decreases the *Sekar–Ramakrishnan measure* [35, 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 4. *A proof term can be transformed into a permutation equivalent greedy multistep reduction.*

Remark 12. *To give an idea how residual theory [35, Table 8.5 in Section 8.7.3] 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 13. *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 [35, 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 [24], 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, see Figure 4,*

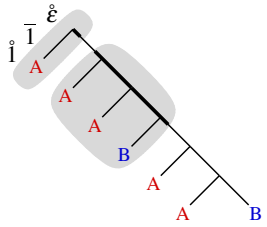


Figure 4: Non-overlapping patterns (in grey) and intervals (thick lines) of A and AAB in $\underline{AAA}BAAB$

using underlining to indicate patterns, that $\alpha AABAAB, A\beta AAB$ in γ is a loath pair follows from that the pattern[13] $\{\mathring{2}, 2 \cdot \overline{1}, 2 \cdot \mathring{1}, 2 \cdot \overline{2}, 2 \cdot \mathring{2}, 2 \cdot 2 \cdot \overline{1}, 2 \cdot 2 \cdot \mathring{1}, 2 \cdot 2 \cdot \overline{2}, 2 \cdot 2 \cdot \mathring{2}, 2 \cdot 2 \cdot 2 \cdot \overline{1}, 2 \cdot 2 \cdot 2 \cdot \mathring{1}\}$ *in $\underline{AAA}BAAB$ corresponding to the source AAB of the rule β, does not have overlap with the pattern $\{\mathring{\varepsilon}, \overline{1}, \mathring{1}\}$ in $\underline{AAA}BAAB$ corresponding to the target A of the rule α. This in turn follows from that the corresponding intervals $[\mathring{2}, 2 \cdot 2 \cdot \mathring{2}]$ and $[\mathring{\varepsilon}]$ are disjoint since $\mathring{\varepsilon}$ 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. *The maps $[\![\,]\!]$ and TS constitute a(n effective) bijection between greedy multistep reductions and tragrs.*

Proof. We first show that the topological multi-sorting function $[\![\,]\!]$ maps tragrs not just to multistep reductions but to greedy such. Next we show that when restricting the domain of the interpretation $[\![\,]\!]$ to greedy multistep reductions, $[\![\,]\!]$ and TS are inverse to each other as functions from greedy multistep reductions to tragrs and conversely. From that we conclude as both the functions $[\![\,]\!]$ and TS are effective.

[13] Patterns comprise node (over-ring) and edge (over-bar) positions; cf. [35, Example 8.6.4] for more on their notation.

We 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 γ 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. Remark 13, the corresponding pair $\Phi\cdot\Psi$ of consecutive multisteps is greedy / not loath. Thus, $[\![\,]\!]$ maps to greedy multistep reductions.

- That TS after $[\![\,]\!]$ is the identity on greedy multistep reductions, we show by induction on the length of such a 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 of the tragr $[\![\Phi\cdot\gamma]\!]$ 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.[14] This contradicts the assumed greediness of $\Phi\cdot\gamma$.
- The converse direction, that $[\![\,]\!]$ after TS is the identity on tragrs, follows from Lemma 3. □

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 multi-sort (using TS) yielding the greedy multistep reduction.

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

Proof. Lemma 4 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.. □

Remark 14.
- *The proof only employs one direction (the first item in 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 local 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 multi-sorting.*

[14] 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 13. 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.

5 Conclusions

We have shown that Lévy's notion of permutation equivalence [18] as known from *term rewriting* [35] corresponds, after specialising it to *string rewriting*, to the notion of causal equivalence as employed by Wolfram in his *physics* project [38, 39]. This we achieved by introducing trace graphs, tragrs refining Wolfram's notion of causal graph, as representatives of permutation equivalence classes of reductions. Representing reductions as terms themselves, so-called proof terms [21], 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 that representatives are 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 multi-sorting procedure, showing both maps to be inverse to each other.

The study of *causality* spans all the sciences, cf. [27], hence it is no surprise that it has been discussed and mathematically modelled in many ways; to mention a few [22, 18, 4, 33, 37, 12, 11, 21, 17, 19, 14, 9, 10, 7, 39]. From that point of view our results can be seen as linking models of causality known from *rewriting* [18] (permutation equivalence), *algebra* [7] (greedy multistep reductions) and *physics* [39] (causal graphs), respectively. In general, we think that linking different perspectives on the *same* notion, as we did for causal equivalence here but also before in [35, Chapter 8], is important. Therefore we find it surprising that in the literature mentioned cross-references beyond the borders of the specific field (rewriting, algebra, physics, category theory, proof theory, concurrency theory,...) of a paper, are few and far between. We hope that our short paper can contribute to creating at least some awareness of that, in our opinion unfortunate, situation for causal equivalence, and the interest in overcoming it.

Remark 15. *Not being a physicist, I am not in a position to assess the potential relation of the various causal models to physics, cf. [39, Section 8]. However, I do think it already* methodologically *interesting to see how far one can push causal models, which phenomena can or cannot be reconstructed from them. For example, in [8] it is argued that* time *cannot be reconstructed from purely* causal *models (such as rewriting), as the latter fail to explain* synchronicity.[15] *That makes one wonder whether adding natural structure (think of a notion of strategy or a metric) could overcome that, could make time emergent.*

The concepts and techniques developed and employed here are simple and natural.[16] For instance, topological multi-sorting 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 and natural do we entertain the hope to extend them to more complex cases. In particular, we hope that the results developed here for *string* rewriting can serve as a stepping stone for tackling the problem, left open in [35, Chapter 8] (see Remark 1), of giving a characterisation of permutation equivalence for *term* rewriting[17] by *tracing*. Interesting (classes of) term rewrite systems to target are:

- *Linear* systems. In the first-order case linearity can be brought about by requiring for each rule that every variable occurs either zero or one time in *both* its sides. The characterisation should cover not only string rewrite systems via their *monadic* embedding (see Remark 2), and *chemical* systems [35, Example 8.6.1], but also rules like $x+0\to x$.[18] In the higher-order case [35, Chapter 11] linearity could be brought about by restricting to a *linear* substitution calculus [32].

[15] Roughly: Why are two identical but independent clocks seen to be in the same state?

[16] This could be the reason for the observed disjointedness of the literature on causal equivalence: natural notions are likely to be developed autonomously multiple times. *Therefore* we think it worthwhile to (try to) link such notions.

[17] But also other types of *structured* rewrite systems such as graph rewrite systems come to mind.

[18] I expect the adaptation of the results to first-order linear term rewriting to be largely unproblematic but still interesting. For instance, trace graphs for term rewrite systems cannot be planar, unlike those for string rewrite systems here.

- *Non-linear* systems. As illustrated in the introduction, the problem in the non-linear case is that the replicating effect of a non-linear term rewrite step is not represented in the term structure, so cannot be traced. One could hope this can be overcome by reifying replication (so it becomes *traceable*), by making the *substitution calculus* [26] suitably explicit. *Sharing graphs* as known from the theory of optimal reduction [1, 25] suggest themselves, in both the first- and higher-order cases, with Combinatory Logic and the $\lambda\beta$-calculus, respectively, concrete systems to try potential characterisations on.

All our results are effective and constructive, but we did not study their complexity. However, we do hope that 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].

Acknowledgments We thank Jan Willem Klop for making the initial remark, Nao Hirokawa and the reviewers and participants of Termgraph 2022 in Haifa for feedback, and the reviewers for thorough reading and many helpful comments and suggestions.

References

[1] A. Asperti & S. Guerrini (1998): *The Optimal Implementation of Functional Programming Languages*. Cambridge University Press.

[2] F. Baader & T. Nipkow (1998): *Term Rewriting and All That*. Cambridge University Press, doi:10.1017/CBO9781139172752.

[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] V.A. Gijsbers (2019): *Time and Causality across the Sciences*, chapter 4: On the Causal Nature of Time, p. 49–71. Cambridge University Press, doi:10.1017/9781108592703.004.

[9] 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.

[10] 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.

[11] 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, pp. 395–443. Update of: Call-by-need computations in non-ambiguous linear term rewriting systems, 1979.

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

[13] 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.

[14] 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.

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

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

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

[18] 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`.

[19] 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`.

[20] 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.

[21] 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.

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

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

[24] 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`.

[25] V. van Oostrom, K.J. van de Looij & M. Zwitserlood (2004): *Lambdascope*. In: *ALPS 2004*, p. 9. Available at `http://www.javakade.nl/research/pdf/lambdascope.pdf`.

[26] V. van Oostrom & F. van Raamsdonk (1994): *Weak Orthogonality Implies Confluence: The Higher Order Case*. In: *LFCS'94*, *LNCS* 813, Springer, pp. 379–392, doi:10.1007/3-540-58140-5_35.

[27] J. Pearl (2009): *Causality: models, reasoning, and inference*, 2nd edition. Cambridge University Press, doi:10.1017/CBO9780511803161.

[28] 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.

[29] D. Pous (2010): *Untyping Typed Algebraic Structures and Colouring Proof Nets of Cyclic Linear Logic*. In A. Dawar & H. Veith, editors: *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, *LNCS* 6247, Springer, pp. 484–498, doi:10.1007/978-3-642-15205-4_37.

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

[31] M. Shanahan (2016): *The Frame Problem*. In E.N. Zalta, editor: *The Stanford Encyclopedia of Philosophy*, Spring 2016 edition, Metaphysics Research Lab, Stanford University.

[32] C.L. Smith (2017): *Optimal Sharing Graphs for Substructural Higher-order Rewriting Systems*. Ph.D. thesis, University of Kent. Available at `https://kar.kent.ac.uk/63884/`.

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

[34] 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.

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

[36] A. Visser (2011): *On the ambiguation of Polish notation*. *Theoretical Computer Science* 412(28), pp. 3404–3411, doi:10.1016/j.tcs.2011.01.025. Festschrift in Honour of Jan Bergstra.

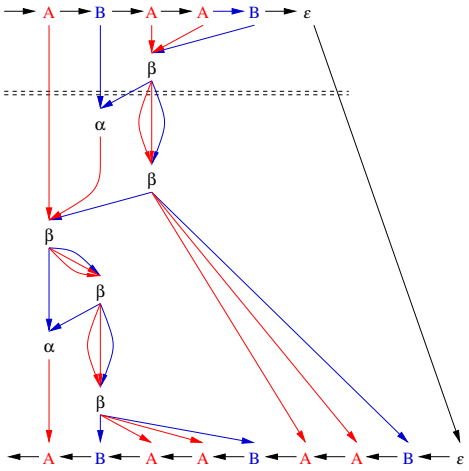
[37] G. Winskel (1989): *An introduction to event structures*. In: *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, *LNCS* 354, Springer, pp. 364–397, doi:10.1007/BFb0013026.

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

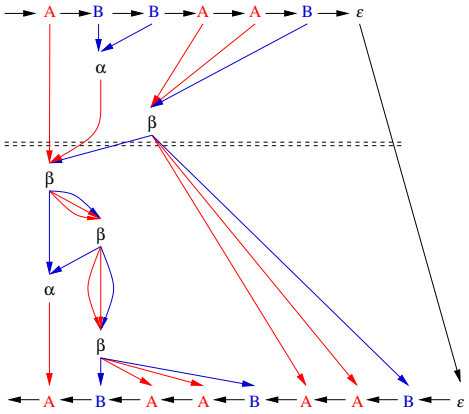
[39] S. Wolfram (2020): *A Class of Models with the Potential to Represent Fundamental Physics*. *Complex Systems* 28(2), pp. 107–536, doi:10.25088/ComplexSystems.29.1.2.

A Topologically multi-sorting the tragr of Figure 2 (right)

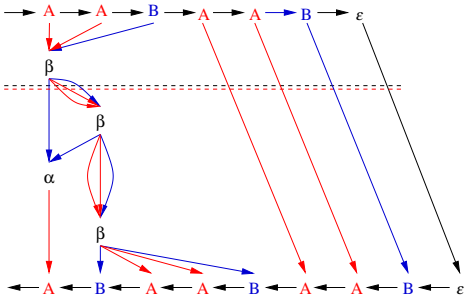
Reading back the tragr in Figure 2 (right) by topological multi-sorting gives rise to the following 6 stages.



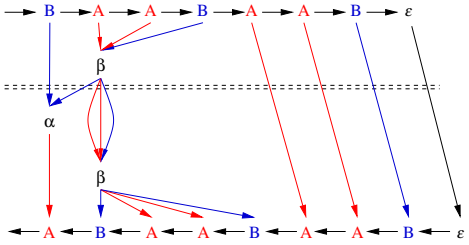
1st multistep: $AB\beta$



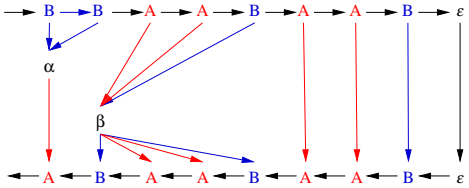
2nd multistep: $A\alpha\beta$



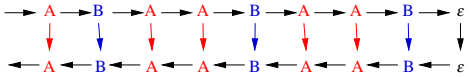
3rd multistep: βAAB



4th multistep: $B\beta AAB$



5th multistep: $\alpha\beta AAB$



Last (empty) multistep: $ABAABAAB$

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' from the joint tragr in Figure 2, of γ and γ' in Example 2!