Graph Path Orderings

Nachum Dershowitz, Jean-Pierre Jouannaud

Graph Path Orderings

Nachum Dershowitz[1] and Jean-Pierre Jouannaud[2,3]

[1] School of Computer Science, Tel Aviv University, Tel Aviv, Israel
nachum.dershowitz@cs.tau.ac.il
[2] LIX, École Polytechnique, Palaiseau, France
[3] Project Dedukti, INRIA-Saclay, France
jeanpierre.jouannaud@gmail.com

Abstract

We define well-founded rewrite orderings on graphs and show that they can be used to show termination of a set of graph rewrite rules by verifying all their cyclic extensions. We then introduce the *graph path ordering* inspired by the recursive path ordering on terms and show that it is a well-founded rewrite ordering on graphs for which checking termination of a finite set of graph rewrite rules is decidable. Our ordering applies to arbitrary finite, directed, labeled, ordered multigraphs, hence provides a building block for rewriting with graphs, which should impact the many areas in which computations take place on graphs.

1 Introduction

In the introduction to their book on the algebra of operads [6], Bremmer and Dotsenko write:

> Elements of algebras are trees. Elements of operads are conventionally represented by linear combinations of trees, "tree polynomials". Generalizations to algebraic structures where monomials are graphs that possibly have loops and are possibly disconnected, e.g. properads, PROPs, wheeled operads, etc., are still unknown, and it is not quite clear if it is at all possible to extend Gröbner-flavored methods to those structures.

We are accordingly interested in well-founded orderings that can be used to show termination of rewriting on first-order terms having both sharing and back-arrows, which is another way of saying that we study rewriting of rooted (multi-) graphs, each vertex of which is labeled by a function symbol, the arity of which governs the number of vertices it points to. Different target applications require different properties of the ordering: totality is crucial for operaders, while monotonicity with respect to graph structure is important for rewriters.

Graph rewriting has been richly studied. Path orderings of term graphs were developed in [9]. Graph decomposition and weight-based orderings for cyclic graphs are explored in [1]. The very successful use of matrix interpretations has been generalized to graph rewriting in [2].

By adding a root structure to cyclic graphs, we get a natural decomposition into head and tail and a concomitant path ordering which generalizes the recursive path ordering to ordered, labeled, rooted graphs, we call them drags. The graph ordering we end up with, *GPO*, has the very same definition as the recursive path ordering (RPO) [3], but the computation of the "head" and "tail" of a multigraph shares little resemblance with the case of trees, for which the head is the top function symbol labeling the root of the tree, and the tail is the list of its subtrees. GPO has many of the properties that are important for its various potential users. It is well-founded, total on graph expressions up to isomorphism, and reduction is testable.

Drags, structural properties of drags, and drag rewriting are reviewed in Sect. 2; see [5] for more details. Drag reduction orderings are then introduced in Sect. 3, and GPO in Sect. 4, its properties in Sect. 5, and ordering drag heads in Sect. 6. Some issues are briefly discussed in the concluding section.

2 The Algebra of Finite, Directed, Labeled Multigraphs

The class of graphs with which we deal here is that consisting of finite directed multi-rooted graphs with labeled vertices and allowing multiple edges between vertices. We assume that the outgoing neighbors (vertices at the other end of outgoing edges) are ordered (from left to right, say) and that their number is fixed, depending solely on the label of the vertex: we presuppose a set of function symbols Σ, whose elements $f \in \Sigma$ used as labels are equipped with a fixed arity, and a denumerable set of variable symbols Ξ disjoint from Σ, whose arity is 0.

We shall call these finite ***d***irected ***r***ooted l***a***belled ***g***raphs, ***drag***s.

Drags have recently been introduced in their full generality in [4, 5]. This section presents the main concepts that are developed there, in particular, drag composition and the associated algebraic structure, the dag decomposition of a drag, and, lastly, drag rewriting.

2.1 Drags

To ameliorate notational burden, we will use vertical bars $|\cdot|$ to denote various quantities, such as length of lists, size of sets or of expressions, and even the arity of function symbols. We use $\varnothing$ for an empty list, set, or multiset, $\cup$ for set and multiset union, as well as for list concatenation, and $\setminus$ for set or multiset difference. We mix these, too, and denote by $K \setminus V$ the sublist of a list K obtained by filtering out those elements belonging to a set V. We will also identify a singleton list, set, or multiset with its contents to avoid unnecessary clutter.

Definition 1 (Drags [5])**.** *A* drag *is a tuple* $\langle V, R, L, X, S\rangle$*, where*

1. V *is a finite set of vertices;*
2. $R : 1 \,..\, |R| \to V$ *is a finite list of vertices, called* roots*, so* $R(n)$ *refers to the* n*th root in the list;*
3. $S \subseteq V$ *is a set of* sprouts*, leaving* $V \setminus S$ *to be the* internal *vertices;*
4. $L : V \to \Sigma \cup \Xi$ *is the* labeling *function, mapping internal vertices* $V \setminus S$ *to labels from the vocabulary* Σ *and sprouts* S *to labels from the vocabulary* Ξ*;*
5. $X : V \to V^*$ *is the* successor *function, mapping each vertex* $v \in V$ *to a list of vertices in* V *whose length equals the arity of its label (that is,* $|X(v)| = |L(v)|$*).*

A drag is closed *if it has no sprouts* S*, and* open *otherwise. The last component* $S = \varnothing$ *will often be omitted from the tuple for closed drags. Open drags act as* patterns*.*

If $b \in X(a)$*, then* (a, b) *is a directed* edge *with* source a *and* target b*. We also write* aXb*. The reflexive-transitive closure* X^* *of the relation* X *is called* accessibility*. A vertex* v *is said to be* accessible from *vertex* u*, and likewise that* u accesses v*, if* uX^*v*. Vertex* v *is* accessible *(without qualification) if it is accessible from some root. A root* r *is* maximal *if any root that can access it is accessible from it. We denote by* $\mathcal{R}^\bullet(R)$ *the set of maximal roots of* R*.*

A drag is: clean *if all its vertices are accessible;* cyclic *if every internal vertex can access some maximal root;* linear *if no two sprouts have the same label.*

The labeling function extends to lists, sets, and multisets of vertices in the expected manner.

Terms as ordered trees, sequences of terms, terms with shared subterms, and drags without backarrows – also called *jungles* [7] – are all drags. More generally, any labeled multigraph can be equipped with a distinguished list of its vertices declared to be its list of roots, turning it into a closed drag. This shows the generality of the notion of drag.

It will sometimes be convenient to consider roots as specific incoming edges and to identify a sprout with the variable symbol that is its label. We use these facilities unannounced.

The component R of a drag is a list of roots possibly with repetitions, whereas S is a set of sprouts. Having a list of roots, rather than a set, is one of the keys to a nice algebra of drags and hence, for a general notion of rewriting. Sprouts may be roots, as we shall see in examples.

Given a drag $D = \langle V, R, L, X, S\rangle$, we make use of the following notations: $\mathcal{V}er(D) = V$ for its set of vertices; $X_D = X$ for its successor function; $\mathcal{A}cc(D) = X^*(R)$ for its set of accessible vertices; $\mathcal{R}(D)$ for its set of roots; $r \in R$ for $\exists n.r = R(n)$; $[1\,..\,|R|]$ for the numbers of the roots in order; $\mathcal{S}(D) = S$ for its set of sprouts; $\mathcal{V}ar(D) = L(S)$ for the set of variables labeling its sprouts; and $D^\sharp$ for the clean drag obtained by removing inaccessible vertices and their related edges, which fits with the underlying intended behavioral semantics of drags.

Drags are graphs. An isomorphism between two open drags is a one-to-one mapping between their respective sets of (accessible) vertices that identifies their respective labels (up to renaming of the sprouts' labels done here by the very same mapping) and lists of roots, and commutes with their respective successor functions. It is sometimes useful to consider multisets instead of lists of roots, resulting in a coarser equivalence on drags: We write $D =_o D'$, or simply $D = D'$ when D and D' are isomorphic drags. We write $D \simeq_o D'$, or simply $D \simeq D'$, and say that D, D' are *quasi-isomorphic* if o restricts to a bijection between the multisets (instead of lists) of roots. We write $D \equiv D'$ and $D \cong D'$ instead of $D = D'$ and $D \simeq D'$ in case o is the identity.

2.2 Composition of Drags

The key to working with drags is that we can equip them with a parameterized binary composition operator that connects sprouts of each of two drags with roots of the other according to a device we call a *switchboard*.

Denote by $\mathcal{D}om(\xi)$ and $\mathcal{I}m(\xi)$ the *domain (of definition)* and *image* of a (partial) function ξ, using $\xi_{A\to B}$ for its restriction going from $A \subseteq \mathcal{D}om(\xi)$ to $B \subseteq \mathcal{I}m(\xi)$, omitting $\to B$ when irrelevant.

Definition 2 (Switchboard)**.** *Let $D = \langle V, R, L, X, S\rangle$ and $D' = \langle V', R', L', X', S'\rangle$ be two open drags. A* switchboard *$\xi : S \cup S' \to \mathbb{N}$ for D, D' splits into a pair $\langle\, \xi_D : S \to [1\,..\,|R'|]$, $\xi_{D'} : S' \to [1\,..\,|R|]\,\rangle$ of partial injective functions called* context *and* substitution*, respectively, such that*

(i) $\forall s, t \in S.\ s \in \mathcal{D}om(\xi_D)$ *and* $L(s) = L(t)$ *imply* $t \in \mathcal{D}om(\xi_D)$ *and* $R'(\xi_D(s)) = R'(\xi_D(t))$*;*

(ii) $\forall s, t \in S'.\ s \in \mathcal{D}om(\xi_{D'})$ *and* $L'(s) = L'(t)$ *imply* $t \in \mathcal{D}om(\xi_{D'})$ *and* $R(\xi_{D'}(s)) = R(\xi_{D'}(t))$*;*

(iii) well-behavedness*: ξ does not induce any cycle among sprouts:*

$$\nexists\, n > 0, s_1, \ldots, s_{n+1} \in S, t_1, \ldots, t_n \in S', s_1 = s_{n+1}.\ \forall i \in 1\,..\,n.\ s_i\ \xi_D\, X'^*\ t_i\ \xi_{D'}\, X^*\ s_{i+1}$$

where we denote by ξ_D the relation on $S \times \mathcal{R}(D')$ defined as $r\,\xi_D\,r$ iff $r = R'(\xi_D(s))$. We will also say that $D'\xi$ is an extension *of D.*

Both conditions (i,ii) are satisfied for linear drags, or for nonlinear drags whose switchboard, called *linear*, is defined for sprouts whose variables are all different. More general conditions, based on drag isomorphism, are given in [5].

Injectivity of ξ and roots being lists with repetitions go along together: switchboards can be seen as directed channels connecting *one* sprout to *one* root. Injectivity implies that the list $\xi_D(\mathcal{D}om(\xi_D))$ is a set, making the set difference $[1\,..\,|R'|] \setminus \xi_D(\mathcal{D}om(\xi_D))$ well defined.

A switchboard induces a binary operation on open drags. The essence of this definition is that the (disjoint) union of the two drags is formed, but with sprouts in the domain of the switchboards merged with the roots referred to in the switchboard images. Of course, one needs to worry about the case where multiple sprouts are merged successively, when the switchboards map sprout to rooted-sprout to rooted-sprout until an internal vertex is eventually obtained, thanks to well-behavedness.

Definition 3 (Target)**.** *Let $D = \langle V, R, L, X, S\rangle$ and $D' = \langle V', R', L', X', S'\rangle$ be drags such that $V \cap V' = \varnothing$, and ξ be a switchboard for D, D'. The* target *$\xi^*(s)$ of a sprout $s \in S \cup S'$ is a vertex v defined as follows: Let $v = R'(n)$ if $s \in S$ and $v = R(n)$ if $s \in S'$, where $n = \xi(s)$. If $v \in \mathcal{D}om(\xi)$, then $\xi^*(s) = \xi^*(v)$, otherwise, $\xi^*(s) = v$.*

$\xi^*(\_)$ is extended to all vertices by letting $\xi^*(v) = v$ when $v \in (V \setminus S) \cup (V' \setminus S')$.

We are ready for defining the composition of two drags.

Definition 4 (Composition)**.** *Let $D = \langle V, R, L, X, S\rangle$ and $D' = \langle V', R', L', X', S'\rangle$ be drags such that $V \cap V' = \varnothing$, and ξ be a switchboard for D, D'. Their* composition *is the drag $D \otimes_\xi D' \stackrel{\text{def}}{=} \langle V'', R'', L'', X'', S''\rangle$, where*

(1) $V'' = (V \cup V') \setminus \mathcal{D}om(\xi)$;

(2) $S'' = (S \cup S') \setminus \mathcal{D}om(\xi)$;

(3) $R'' = \xi^*(R([1\,..\,|R|] \setminus \xi_{D'}(\mathcal{D}om(\xi_{D'})))) \cup \xi^*(R'([1\,..\,|R'|] \setminus \xi_D(\mathcal{D}om(\xi_{D'}))))$;

(4) $L''(v \in V \cap V'') = L(v)$ *and* $L''(v \in V' \cap V'') = L'(v)$;

(5) $X''(v \in V \setminus S) = \xi^*(X(v))$ *and* $X''(v \in V' \setminus S') = \xi^*(X'(v))$.

It is easy to see that $D \otimes_\xi D'$ is a well-defined drag, that is, it satisfies the arity constraint required at each vertex. Note also that, if ξ_D is surjective and $\xi_{D'}$ total, a category of switchboard that will play a key role for rewriting, then *all* roots and sprouts of D' disappear in the composed drag (if $\xi_{D'}$ is surjective and ξ_D total, those from D disappear). Otherwise, the symmetry of the definition is broken by choosing the roots originating from D to come first.

Example 5. *We show in Figure 1 three examples of compositions, the first two with similar drags. The first composition is a substitution of terms. The second uses a bi-directional switchboard which induces a cycle. In the second example, the remaining root is the first (red) root of the first drag which has two roots, the first red, the other black. The third example shows how sprouts that are also roots connect to roots in the composition (colors indicate roots' origin). Note that the root number 3 of the right-hand side drag has disappeared in the composition, while its root number 2 is now root number 4 of the result. This agrees with the definition, as shown by the following calculations (naming vertices by their label without ambiguity in this example):* $\xi^*(x) = \xi^*(y) = h$; $R : [1..3] \to \{f, h, x\}$, $R' : [1..3] \to \{g, y\}$, $R'' : [1..4] \to \{f, g, h\}$; $R'' = \xi^*(R([1\,..\,3] \setminus 2)) \cup \xi^*(R'([1\,..\,3] \setminus 3)) = \xi^*(f, x) \cup \xi^*(g, y) = f\ h\ g\ h$.

Composition has three important algebraic properties [5], which we describe in turn.

Our definition of switchboard being (almost) symmetric, composition is itself symmetric provided all roots of the resulting drag originate from the same side. Otherwise, composition yields drags that are equal up to cyclic permutation of their roots.

Composition has identities: the *identity extension* of D is the pair $1^X_X\iota$ made of an identity drag 1^X_X with no internal vertices nor edges and a set of sprouts $X \subseteq \mathcal{V}ar(D)$. An *identity* switchboard ι is such that $\mathcal{D}om(\iota_D) = X$, ι_D is the identity (abusing our notations) and $\mathcal{D}om(\iota_{1^X_X}) = \varnothing$. We use $\varnothing$ for the drag $1^\varnothing_\varnothing$, called the *empty drag.*

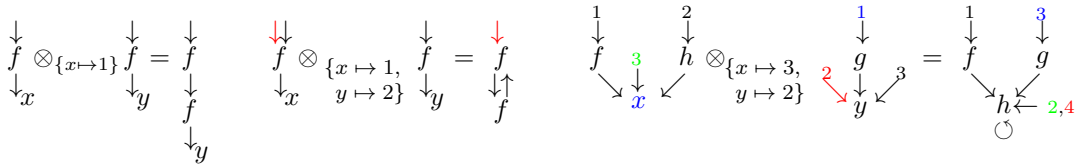

Figure 1: Unidirectional, cyclic and root-transfer compositions.

Composition is associative: let U, V, W be three drags, and ξ, ζ be two switchboards for U, V and V, W, respectively, such that $\mathcal{D}om(\xi_V) \cap \mathcal{D}om(\zeta_V) = \varnothing$ and $\mathcal{I}m(\xi_U) \cap \mathcal{I}m(\zeta_W) = \varnothing$. Then, $(U \otimes_\xi V) \otimes_\zeta W = U \otimes_\xi (V \otimes_\zeta W)$.

2.3 Drag Rewriting

Definition 6 (Rules)**.** *A* graph rewrite rule *is a pair of open clean drags, written $L \to R$, such that $|\mathcal{R}(L)| = |\mathcal{R}(R)|$, $\mathcal{V}ar(R) \subseteq \mathcal{V}ar(L)$, and L is not a variable.*

A graph rewrite system is a set of graph rewrite rules.

Trees are particular clean drags. Because they have a single root, term rewrite rules satisfy the first condition. The second must be explicitly stated in the definition of a term rewrite rule, as for drags.

Definition 7. *Given an open drag L, we say that the extension $C\xi$ of L is a rewriting extension if ξ_C is surjective and ξ_L is total.*

Definition 8 (Rewriting)**.** *Let G be a graph rewrite system. We say that a nonempty clean drag D* rewrites *to a clean drag D', and write $D \longrightarrow_G D'$ iff $D = C \otimes_\xi L$ and $D' = (C \otimes_\xi R)^\sharp$ for some drag rewrite rule $L \to R \in G$ and rewriting extension $C\xi$ of L.*

All assumptions on ξ play an essential rôle. First, because ξ is a rewriting switchboard, ξ_C must be linear. This implies that the variables labeling the sprouts of C which are not already sprouts of D must be all different. Second, ξ_C must be surjective, implying that the roots of L disappear in the composition. Third, ξ_L must be total, implying that the sprouts of L disappear in the composition. Fourth, D must be nonempty, implying that the roots of W do not all disappear in the composition; hence ξ_L cannot be surjective.

What is remarkable about the above definition is that there is no longer any distinction between the context and substitution of a rewrite, both constitute the open drag rewriting extension of the left-hand side. It also explains why the two components of a switchboard were named context and substitution, respectively.

Example 9. *Consider the second composition of Figure 1, and let $f(y) \to y$ be a graph rewrite rule whose left-hand and right-hand sides are, more precisely, the drags $\langle\{f_1, y_2\}, f_1, \{f_1 \mapsto_L f, y_2 \mapsto_L y\}, f_1 \mapsto_X y_2, y_2\rangle$ and $\langle y_1, y_1, y_1 \mapsto_L y, \varnothing, y_1\rangle$, respectively. Then, the drag $\langle\{f_1, f_2\}, f_1, \{f_1, f_2 \mapsto_L f\}, \{f_1 \mapsto_X f_2, f_2 \mapsto_X f_1\}\rangle$ rewrites to the drag $\langle\{f_1, x_2\}, f_1, \{f_1 \mapsto_L f, x_2 \mapsto_L x\}, f_1 \mapsto_X x_2, x_2\rangle \otimes_{\{x_2 \mapsto 1, y_1 \mapsto 1\}} \langle y_1, y_1, y_1 \mapsto_L y, \varnothing, y_1\rangle = \langle f_1, f_1, f_1 \mapsto_L f, f_1 \mapsto_X f_1\rangle$: a cycle of length 2 has been rewritten into one of length 1.*

It is also possible to break a cycle in a drag: the same drag $\langle\{f_1, f_2\}, f_1, \{f_1, f_2 \mapsto_L f\}, \{f_1 \mapsto_X f_2, f_2 \mapsto_X f_1\}\rangle$ rewrites to $\langle\{f_1, x_2\}, f_1, \{f_1 \mapsto_L f, x_2 \mapsto_L x\}, f_1 \mapsto_X x_2, x_2\rangle \otimes_{x_2 \mapsto 1} \langle a_1, a_1, a_1 \mapsto_L a, \varnothing\rangle = \langle\{f_1, a_2\}, f_1, \{f_1 \mapsto_L f, a_2 \mapsto_L a\}, f_1 \mapsto_X a_2\rangle$ with the rule $f(y) \mapsto a$ both sides of which are the drags $\langle\{f_1, y_2\}, 1, \{f_1 \mapsto_L f, y_2 \mapsto_L y\}, f_1 \mapsto_X y_2, y_2\rangle$ and $\langle a_1, a_1, a_1 \mapsto_L a, \varnothing\rangle$.

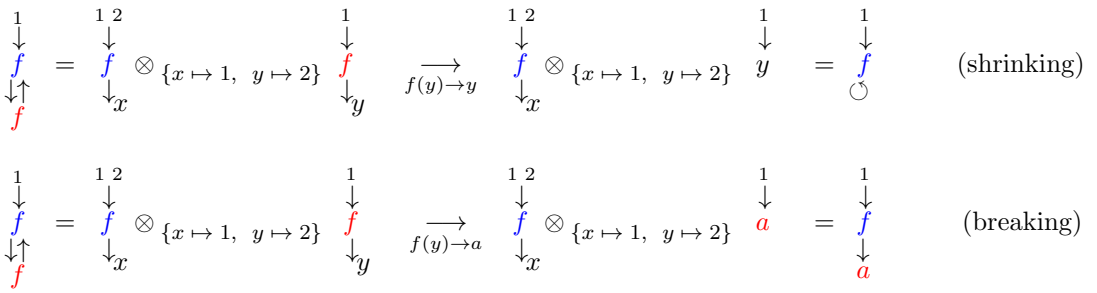

Figure 2: Rewriting a cycle.

These rewrites are shown in Figure 2. In both cases, the upper occurrence of f (in blue) is part of the context, while the one below (in red) is part of the rewrite rule. Rewriting the upper occurrence of f instead of the lower is an exercise left to the reader.

Note that the drag reduced to a loop on the rooted vertex f would not rewrite with any rule of left-hand side $f(y)$: matching would require an ill-formed switchboard (see [5]).

Lemma 10. *If $U \longrightarrow_R V$, then $\mathcal{V}ar(V) \subseteq \mathcal{V}ar(U)$.*

Lemma 11. *Assume U, V, W are three open drags such that the pair (U, V) is a rewrite and ξ is a switchboard for W, U. Then, ξ is a switchboard for W, V.*

2.4 Structure of Rewriting Extensions

First, we categorize extensions according to their impact on the drag they connect to.

Definition 12 (Categories of Extensions)**.** *An extension $W\xi$ of a clean drag U is called*
(i) a context extension *if $\mathcal{D}om(\xi_U) = \varnothing$;*
(ii) a substitution extension *if $\mathcal{D}om(\xi_W) = \varnothing$;*
(iii) a directed extension *if it is either a context extension or a substitution extension;*
(iv) a cyclic extension *if ξ_U is onto $\mathcal{R}^\bullet(W)$, ξ_W is linear, and $W \otimes_\xi U$ is a clean nonempty drag, all of whose internal vertices access some sprout in $\mathcal{D}om(\xi_W)$.*

The rôle of substitution extensions is to modify the structure of U by introducing sharing among its sprouts. Particular substitution extensions are identity substitution extensions $1^z_z\xi$ such that $\xi(x_1) = \cdots = \xi(x_n) = z$, where $x_1, \ldots, x_n$ are variables labeling sprouts of U. Then, the resulting drag is the same as U, except that all its sprouts labeled by the variables $x_1, \ldots, x_n$ are now merged into a single sprout labeled by z.

The rôle of cyclic extensions is to modify the structure of U without changing its internal vertices by connecting some of its sprouts to some of its roots. Particular cyclic extension of U are *identity cyclic extensions*, of the form $1^Z_Y\iota$, where the variables in Y are one-to-one with those in some $X \subseteq \mathcal{V}ar(U)$, $Y \subseteq \mathcal{I}m(\iota_U)$ and $\iota_1 : Y \to [1 \,..\, |R|]$ is an arbitrary map. If the drag U has a single root, identity extensions are enough to predict all shapes that a drag may take under composition with an extension. This is no longer true with multi-rooted drags, since identity extensions cannot reach two different roots from the same sprout.

Notice that the only possible cyclic extension for a tree, dag and jungle is the identity.

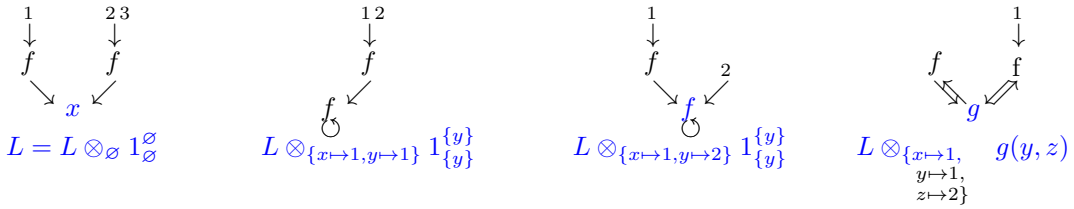

Figure 3: Cyclic extensions (roots of the resulting drags are relocated).

Example 13. *Let L be the drag made of two copies of the tree $f(x)$ sharing the variable x. Three identity extensions of L are represented at Figure 3. The first is the trivial directed extension, while the two others are cyclic ones. The second maps the variable x of L to the only root of the identity drag, which is its sprout y, and the only sprout y of the identity drag to the first root of L. The third maps instead y to the second root. Note that no identity cyclic extension can cover all roots at the same time. No cyclic extension can either if the vocabulary does not contain a binary symbol: a cyclic extension must have a single vertex as root, possibly repeated, since L has a single sprout, and all internal vertices of the extension must be accessible from that sprout. Then, due to the vocabulary, the extension will cover a single root of L.*

Next, we give construction which splits a drag into a *subdrag* generated by some set of vertices, and the rest of the drag, its *antecedent*.

Definition 14 (Subdrag)**.** *Given a drag $U = \langle V, R, L, X, S\rangle$, and a list of vertices W such that each $w \in W$ occurs as many times in W as the number of incoming edges to w in U, the* subdrag $U_{|W}$ *of U generated by W is the drag $\langle V', R', L, X, S'\rangle$, where*

(i) V' is the least superset of W that is closed under X;

(ii) L', X', S' are the restrictions of L, X, S to V';

(iii) R' is $(R \cap V') \cup X(V \setminus V')$.

Its roots are obtained by adding as new roots those vertices which have an incoming edge in U that is not in $U_{|W}$. The order of elements in this additional list does not matter here.

Lemma 15 (Antecedent)**.** *Given a drag D and a list of vertices $W \subseteq V$, there exists a drag A, called* antecedent*, and a linear directed switchboard ξ such that $D = A \otimes_\xi D_{|W}$.*

The fact that the switchboard ξ is directed expresses the property that decomposing a drag into a subdrag and its antecedent does not break any of its cycles. Note further that ξ implicitly defines an order on the roots of the subdrag which are not roots of the whole drag.

We are now ready for characterizing the structure of surjective (hence, rewriting) extensions:

Lemma 16 (Decomposition)**.** *Let U be a clean nonempty drag and $W\xi$ an extension of U such that ξ_W is onto $\mathcal{R}^\bullet(U)$. Then, there exists a cyclic extension $B\xi$ of U such that $W \otimes_\xi U = A \otimes_\zeta ((B \otimes_\xi U) \otimes_\theta C)$, where ζ and θ are directed.*

In this decomposition, C is the subdrag of U generated by those vertices that cannot reach a sprout in $\mathcal{D}om(\xi_U)$, B is the antecedent of C in the subdrag of U generated by the roots in $\mathcal{I}m(\xi_L)$, and A is the antecedent of that subdrag in U.

The Decomposition Lemma illustrates the difference between drags on the one hand, and trees, dags or jungles on the other hand. For the latter three, there is no room for non-trivial

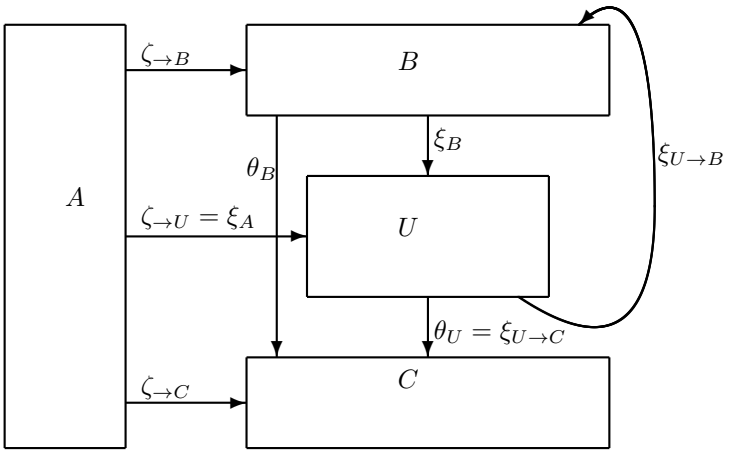

Figure 4: Decomposition of a composition $W \otimes_\xi U$ for which ξ_W is onto $\mathcal{R}^\bullet(U)$.

cyclic extensions, because cycles don't exist. At the same time, it illustrates a resemblance between these different structures: all are compositional. In the case of trees (or dags), compositionality goes without saying: encapsulating a term in a context can be repeated ad libitum, and it is the same for instantiating a term. In the case of drags, the presence of cycles complicates the situation, but our notion of composition remains – at it should – compositional. This is a key property of drags equipped with the family of compositions indexed by switchboards.

Finally, as expected, decomposition is compatible with rewriting:

Lemma 17. *Let $U \longrightarrow_{L\to R} U'$. Then, $U = A \otimes_\zeta ((B \otimes_\xi L) \otimes_\theta C)$ and $U' \equiv A \otimes_\zeta ((B \otimes_\xi R) \otimes_\theta C)$ for some A, B, C and ζ, ξ, θ such that*

1. *$C\theta$ is a substitution extension for both $B \otimes_\xi L$ and $B \otimes_\xi R$;*
2. *$A\zeta$ is a context extension for both $(B \otimes_\xi L) \otimes_\theta C$ and $(B \otimes_\xi R) \otimes_\theta C$;*
3. *$B\xi$ is a cyclic extension for both L and R if $\mathcal{V}ar(R) = \mathcal{V}ar(L)$. If $\mathcal{V}ar(R) \subsetneq \mathcal{V}ar(L)$, then $B = A' \otimes_\delta B'$ for some drags A', B' such that $B'\xi$ is a cyclic extension of R and $A'\delta$ is a context extension of $B' \otimes_\xi R$.*

3 Drag Orderings

We now consider the properties that drag orderings need to satisfy for proving termination.

Definition 18 (Reduction Ordering)**.** *A* (graph) reduction ordering *is a well-founded ordering $\succ$ of the set of drags that is*
(i) compatible *with isomorphism: if $D \succ E$, $D \cong D'$ and $E \cong E'$, then $D' \succ E'$;*
(ii) monotonic*: if $D \succ E$ and $A\xi$ is a context extension of D, then $A \otimes_\xi D \succ A \otimes_\xi E$;*
(iii) stable*: if $D \succ E$ and $C\xi$ is a substitution extension of D, then $D \otimes_\xi C \succ E \otimes_\xi C$.*

Apart from compatibility with isomorphism, the notion of reduction ordering is the same as the usual one. Monotonicity is the usual property since the associated directed switchboard

turns the context A into a usual context. Stability corresponds to the usual stability property for the same reason, but substitution extensions can now introduce sharing.

Cyclic extensions do not show up in this definition because they are not relative to the reduction relation, but rather to the rewrite rules themselves, as we see next:

Theorem 19 (Termination)**.** *A graph rewrite system $\mathcal{R}$ terminates iff there's a graph reduction ordering $\succ$ such that, for all rules $G \to G' \in \mathcal{R}$ and for all cyclic extensions $B\xi$ of G, we have $B \otimes_\xi G \succ B \otimes_\xi R$.*

Proof. Assume $U \longrightarrow_{\mathcal{R}} V$. By Lemma 17, $U = A\otimes_\zeta((B\otimes_\xi L)\otimes_\theta C)$ and $V = A\otimes_\zeta((B\otimes_\xi R)\otimes_\theta C)$ such that $A\zeta$, $C\theta$ and $B\xi$ are context, substitution, and cyclic extensions, respectively. By assumption, $B \otimes_\xi L \succ B \otimes_\xi R$. By stability, $(B \otimes_\xi L) \otimes_\theta C \succ (B \otimes_\xi R) \otimes_\theta C$. By monotonicity, $A \otimes_\zeta ((B \otimes_\xi L) \otimes_\theta C) \succ A \otimes_\zeta ((B \otimes_\xi R) \otimes_\theta C)$. The result now follows from well-foundedness of $\succ$ and its compatibility with drag isomorphism.

For the converse, the derivation relation itself defines a graph rewrite ordering for R. □

The rôle of cyclic extensions in the above test is to check monotonicity of the graph reduction ordering with respect to those cycles that are reducible by R: building this monotonicity property in the definition of a graph reduction ordering would be too strong. In the case of trees, dags and jungles, left-hand sides of rules have no cyclic extensions, explaining why a single test suffices for each rule in these three cases. More generally, we suspect that the case of uni-rooted drags should be much easier than the general case explored here, since identity cyclic extensions should then be able to mimic arbitrary cyclic extensions. We have not explored that path.

4 Path Orderings

A tree headed by the symbol f of arity n has one root labeled by f, or equivalently a head corresponding to the expression $f(x_1, \ldots, x_n)$, and several subtrees, seen here as a single drag with several roots. A major component of RPO is then a well-founded order on heads, that is on function symbols in that case. We therefore start defining a notion of head for an arbitrary drag, before to define a path ordering for drags. Since heads and tails play a major rôle for pattern matching drags, they have already been introduced and studied in [5].

4.1 Heads and Tails

A drag D has indeed a head H and a tail T such that $D = H \otimes_\xi T$ for some linear directed switchboard ξ.

Definition 20 (Head and Tail)**.** *The* head $\widehat{D}$ *of a drag $D = \langle V, R, L, X, S\rangle$ is the drag whose vertices are those of the largest cycle containing the maximal roots of the drag, plus additional sprouts in $\mathcal{D}om(\xi_{\widehat{D}})$ that serve for connecting the head with the tail, whenever the latter is nonempty, by means of a linear switchboard ξ. Its* tail ∇D *is the subdrag generated by the set $V \setminus \{v \in V : vX^*\mathcal{R}^\bullet(R)\}$, made of the vertices of D that cannot reach a maximal root. It may be a list of several distinct connected components.*
We define $\mathcal{V}ar_H(D)$ to be $\mathcal{V}ar(D) \cap \mathcal{V}ar(\widehat{D})$, those variables of D that remain in the decapitated head.

Lemma 15 shows that the head is the antecedent of the tail. A drag can therefore be seen as a bipartite graph made of its head, its tail, and a linear directed switchboard specifying that correspondence.

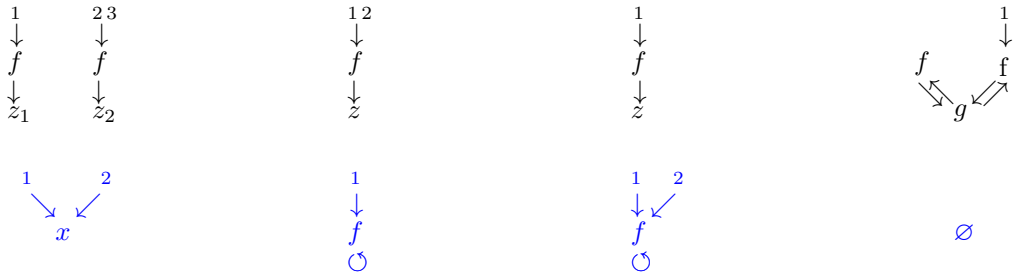

Figure 5: Heads (above) and tails (below in blue) of the drags of Figure 3.

Example 21. *The heads (in black) and tails (in blue) of the four drags pictured at Figure 3 are listed at Figure 5. Their linear connecting switchboards can be easily inferred from the incoming arrows of the head. Linearity imposes that the first head has two sprouts, labeled z_1 and z_2, and the tail has then two roots, number 1 and 2, the switchboard connecting z_1 to root 1 and z_2 to root 2. For the third example, the tail has one root (number 2) inherited from the original drag because it was non-maximal, the variable z from the head mapping to the root number 1.*

In the sequel, we assume that the additional sprouts of the head are ordered with respect to a depth-first search initialized with its list of roots. This order on the additional sprouts of the head induces, via the switchboard ξ, an order on the roots of the associated tail which were not also roots of the original drag. Consequently, the tail is now canonically determined.

Theorem 22 (Structure [5])**.** *Isomorphic drags have isomorphic heads and tails.*

The fundamental property of a drag expressed by this theorem is that its decomposition into a head and tail is a faithful representation of that drag. This property holds true because drags are multi-rooted.

4.2 GPO

We now define GPO, a family of graph rewrite orderings. To this end, we first introduce the analog for drags of the precedence on function symbols used by RPO:

Definition 23 (Head Order)**.** *A quasi-order on cyclic drags is said to be*

(i) compatible*: if isomorphic cyclic drags are equivalent in the quasi-order;*

(ii) stable: *if $\widehat{U} \geq \widehat{V}$ implies $\mathcal{V}ar_H(V) \subseteq \mathcal{V}ar_H(U)$;*

(iii) delible: *if $L \otimes_\xi E > R \otimes_\xi E$ implies $L > R$ and $L \otimes_\xi E \doteq R \otimes_\xi E$ implies $L \doteq R$, where L, R are cyclic drags and $E\xi$ is a cyclic extension of both L and R;*

(iv) solvable*: if universal first-order head-order constraints are decidable.*

A head order *for a drag rewrite system $\mathcal{R}$ is a compatible quasi-order $\geq$ on clean cyclic drags whose strict part is well-founded. A* rewrite head order *is a stable head order.*

Compatibility strictly includes cyclic-drag isomorphism. Delibility and solvability play a key rôle for showing the decidability of GPO-based termination proofs. Delibility and totality together imply monotonicity of the head order, that is, its preservation when growing cycles.

Definition 24 (GPO). *Given two drags s, t and a rewrite head order $\succeq$, we define $s \succ t$ under the* graph path ordering (GPO) *iff any of the following holds:*

∇: $\nabla s \succeq t$;

$\succ$: $\widehat{s} \succ \widehat{t}$ *and* $s \succ \nabla t$;

$\doteq$: $\widehat{s} \doteq \widehat{t}$, $s \succ \nabla t$ *and* $\nabla s \succ \nabla t$.

The empty drag is minimal, as we will see.

It is worth noting here that the two drags s, t need not be clean. Cleaning proceeds by comparing heads and tails of the drags, therefore ensuring compatibility of the order with drag isomorphism.

Example 25. *Consider the goal* $\langle\{f_1, f_2\}, f_1, \{f_1, f_2 \mapsto_L f\}, \{f_1 \mapsto_X f_2, f_2 \mapsto_X f_1\}\rangle \succ \langle\{f_1, a_2\}, f_1, \{f_1 \mapsto_L f, a_2 \mapsto_L a\}, f_1 \mapsto_X a_2\rangle$*, which corresponds to the rule of the (breaking) case in Figure 2. The first drag is a head. The head of the second is the drag* $\langle\{f_1, x_2\}, f_1, \{f_1 \mapsto_L f, x_2 \mapsto_L x'\}, f_1 \mapsto_X x', x_2\rangle$*. It is easy to see that the first head can be obtained from the second by composition. We consequently assume that it is larger in the head order. Using Case $\succ$ of GPO, we must now prove the subgoal:* $\langle\{f_1, f_2\}, f_1, \{f_1, f_2 \mapsto_L f\}, \{f_1 \mapsto_X f_2, f_2 \mapsto_X f_1\}\rangle \succ \langle a_1, a_1, a_1 \mapsto_L a, \varnothing\rangle$*. This time, the second drag is a head itself. Therefore, we must have* $\langle\{f_1, f_2\}, f_1, \{f_1, f_2 \mapsto_L f\}, \{f_1 \mapsto_X f_2, f_2 \mapsto_X f_1\}\rangle \succ \langle a_1, a_1, a_1 \mapsto_L a, \varnothing\rangle$ *for this goal to succeed. As we shall see, such a head order is easy to define.*

GPO has exactly the same definition as RPO, the recursive path ordering for terms, dags, or more generally, term graphs. As in those cases, the difference is that the head of a drag has a definition that is specific to its structure, and this is true of the tail as well. Here, this definition has to cope with the existence of cycles, which are banned from term graphs.

5 Properties of GPO

We now prove the main properties of GPO ($\succ$) implying that it is a graph rewrite ordering, with some additional properties that are important in practice. First, GPO is defined by induction on the pair $\langle s, t\rangle$ via the lexicographic extension of the subdrag order:

Let $t \rhd u$ if $u = \nabla t$.

Lemma 26 (Subdrag Order). *The subdrag order $\rhd^+$ is well-founded on nonempty open drags.*

Proof. $\rhd$ decreases the number of vertices of a nonempty drag by removing its head. □

Lemma 27 (Minimality). *The empty drag is minimal in $\succ$.*

Proof. Taking tails repeatedly will eventually lead to an empty drag, so Case ∇ applies to $u \succ \varnothing$ for nonempty u. On the other hand, no case applies meaningfully to $\varnothing \succ v$. □

Lemma 28 (Subterm). $\rhd \subseteq \succ$.

Proof. By Case ∇ of Definition 24, $U \succ \nabla U$. □

Lemma 29 (Variables). *If $U \succ V$, then $\mathcal{V}ar(V) \subseteq \mathcal{V}ar(U)$.*

Proof. This is a well-known property of RPO-like definitions, which – in this case – uses stability of the head order. □

Theorem 30 (Transitivity)**.** *GPO ($\succ$) is transitive.*

The proof is standard.

Theorem 31 (Compatibility)**.** *GPO is a strict ordering compatible with drag isomorphism.*

Proof. Antisymmetry follows from well-foundedeness proved next, and transitivity is Theorem 30. Compatibility follows from Theorem 22, since the definition of the order uses only heads and tails of the drags to be compared. □

This result is of course important since it justifies our way of building compatibility into a path ordering via a definition of subdrags, which is itself compatible, instead of using a normal-form representation of congruence classes of drags.

Let *SN* stand for the strongly normalizing (terminating) property of an element with respect to GPO.

Lemma 32. *If the drag ∇t is SN, then the drag t is SN.*

Proof. The proof follows closely the proof schema initiated in [8]. By definition, t is SN iff all its reducts vis-à-vis $\succ$ are SN. We prove that each given reduct v of t is SN by induction on the triple $\langle t, \nabla t, v\rangle$, compared in the lexicographic composition $\ggg \stackrel{\text{def}}{=} (\succeq, \succ, \rhd)_{lex}$. For this composition to be well-founded, each component relation must be well-founded. First, $\succ$ is well-founded by definition of a head order. Second, $\rhd$ is well-founded by Lemma 26. Third, ∇t is SN by assumption. Therefore, $\ggg$ is well-founded. There are two kinds of reducts whose triples are smaller than $\langle t, \nabla t, v\rangle$: the reducts of t smaller than v in $\rhd$; and the reducts of some s such that the pair $\langle s, \nabla s\rangle$ is strictly smaller than the pair $\langle t, \nabla t\rangle$ in the order $(\succeq, \succ)_{lex}$. By "hypothesis (n)", we will mean to apply the induction hypothesis in the case where the n-th component of the triple has decreased.

We distinguish the three usual cases:

∇: $\nabla t \succeq v$. Since ∇t is SN by assumption, v is SN by definition of the SN predicate.

$\succ$: $t \succ v$ and $t \succ \nabla v$. By definition of $\rhd$, $v \rhd \nabla v$. Therefore ∇v is SN by hypothesis (3). Let now w be an arbitrary reduct of v. Then, w is SN by hypothesis (1). It follows that v is SN by definition.

$\doteq$: $t \doteq v$, $t \succ \nabla v$, and $\nabla t \succ \nabla v$. By the same token as above, ∇v is SN. Let w be an arbitrary reduct of v. Then, w is SN by hypothesis (2). Hence, v is SN by the definition of SN, and we are done. □

Theorem 33 (Well-Foundedness)**.** *GPO is well-founded.*

Proof. This follows from the previous lemma by straightforward induction on $\rhd$. □

Theorem 34 (Totality)**.** *If the head order is total on equivalence classes of cyclic drags modulo isomorphism, then GPO is total on equivalence classes of drags modulo isomorphism.*

Proof. Let t and v be two non-equivalent drags. We prove that they are comparable by induction on the relation $\rhd$. By the induction hypothesis, t and ∇v are comparable, as well as v and ∇t. If $\nabla v \succeq t$ or $\nabla t \succeq v$, then $v \succ t$ or $t \succ v$, respectively, by Case ∇ of GPO. Otherwise, $t \succ \nabla v$ and $v \succ \nabla t$. Since $\succeq$ is total on cyclic drags, there are three cases, of which two are symmetrical. If $\widehat{t} \succ \widehat{v}$ or $\widehat{v} \succ \widehat{t}$, then $t \succ v$ or $v \succ t$, respectively, by Case $\succ$ of GPO. We are therefore left with the case where $\widehat{t} \doteq \widehat{v}$, for which we simply need to show that $\nabla t \succ \nabla v$ or $\nabla v \succ \nabla t$. By the induction hypothesis, ∇t and ∇v are comparable. Since t and v are not equivalent, ∇t and ∇v are not equivalent either by Theorem 22, ensuring that one drag is bigger than the other. □

Theorem 35 (Monotonicity)**.** *GPO is monotonic.*

Proof. Let $L \succ R$ be two drags and $W\xi$ be a context extension of L. By Lemma 10, ξ is also a directed switchboard for W, R.

The proof is by induction on W with respect to $\rhd$. By Theorem 31, we can assume without loss of generality that L, R are clean. If W is an empty drag, the result follows from commutativity of composition, from the fact that $\varnothing$ is an identity element, and from Theorem 31.

If W is nonempty, then $W = \widehat{W} \otimes_\zeta \nabla W$ for nonempty $\widehat{W}$ and ζ is a directed switchboard. By associativity (which applies here), $W \otimes_\xi L = \widehat{W} \otimes_\zeta (\nabla W \otimes_\xi L)$ and $W \otimes_\xi R = \widehat{W} \otimes_\zeta (\nabla W \otimes_\xi R)$. Since $\widehat{W}$ is a head and ζ is directed, it is the head of both $W \otimes_\xi L$ and $W \otimes_\xi R$ by Theorem 22. By the induction hypothesis, $\nabla W \otimes_\xi L \succ \nabla W \otimes_\xi R$. It follows that $W \otimes_\xi L \succ \nabla W \otimes_\xi R$ by case ∇ of GPO. We then conclude that $W \otimes_\xi L \succ W \otimes_\xi R$ by case $\doteq$ of GPO. □

Theorem 36 (Stability)**.** *GPO is stable.*

Proof. Assume that $U \succ V$ and let $C\xi$ be a substitution extension for U. By Lemma 29, it is a substitution extension for V. The proof that $U \otimes_\xi C \succ V \otimes_\xi C$ is by induction on $|U|$ and by cases upon the proof that $U \succ V$.

Let $U = \widehat{U} \otimes_\zeta \nabla U$, and $V = \widehat{V} \otimes_\zeta \nabla V$, where ζ and θ are both directed. Note that $C\xi$ is a substitution extension for both ∇U and ∇V.

1. Case ∇. Then, $\nabla U \succeq V$. By the induction hypothesis, $\nabla U \otimes_\xi C \succeq V \otimes_\xi C$. By the same token as in the monotonicity proof, $\widehat{U}$ is the head of $\widehat{U} \otimes_\xi C$ and $\nabla U \otimes_\xi C$ is its tail. We therefore conclude by Case ∇ of GPO.

2. Case $\succcurlyeq$. Then, $\widehat{U} \geq \widehat{V}$ and $\nabla U \succ V$. By the induction hypothesis, $\nabla U \otimes_\xi C \succ V \otimes_\xi C$. By the same token as before, $\widehat{U}, \widehat{V}$ are the heads of $U \otimes_\xi C$ and $V \otimes_\xi C$, respectively, and $\nabla U \otimes_\xi C$ is the tail of $U \otimes_\xi C$. We can therefore conclude by Case $\succcurlyeq$ of GPO.

3. Case $\doteq$. By the induction hypothesis and similar arguments again. □

All these proofs are reminiscent of those for RPO.

Now comes a crucial property of GPO, one that departs from the usual properties of RPO on terms, dags or jungles (although it relates to the inference of a precedence in this context): the ability to check that $B \otimes_\xi L \succ B \otimes_\xi R$ for all the infinitely many cyclic extensions $B\xi$ of L – there is only one (trivial) extension in all these cases. As can be expected from the solvability property of a head order, this property relies on a reduction to head-order constraints:

Lemma 37 (Reduction)**.** *The formula $\forall M\xi.\ M \otimes_\xi L \succ M \otimes_\xi R$, where $M\xi$ is an arbitrary cyclic extension of L, reduces to an equivalent universal first-order formula over the head-order predicate.*

Proof. The proof proceeds in three stages.

Firstly, the infinite set $M\xi$ of cyclic extensions of L can be partitioned into $|I|$-many subsets $\{\{M_i^j \xi_i\}_j\}_{i \in I}$, with I finite and all infinitely many extensions $M_i^j \xi_i$ in the subsets $\{M_i^j \xi_i\}_j$ sharing essentially the same switchboard ξ_i. This is so, because there are only finitely many possible different switchboards ξ_i for a given open drag L, since the domain of ξ_L and the image of ξ_M are both finite. To get identical switchboards ξ_i for all of $\{M_i^j \xi_i\}_j$ just rename the (linear) variables of M_i^j that are in $\mathcal{D}om(\xi_i)$ to correspond to their target roots of L.

Secondly, we apply Lemma 16 to a given cyclic extension $M\xi$, considering this time $L\xi$ as being an extension of M. The decomposition lemma applies since ξ_L is onto $\mathcal{R}^\bullet(M)$ by the definition of cyclic extensions. We therefore obtain a context extension L', a cyclic extension

L'', and a substitution extension L''', which depend only on L and ξ_M, hence are identical for all extensions in the same subset $\{M_i^j \xi_i\}_j$.

We now claim that $L'' \otimes M$ is a cyclic drag. This is so because every internal vertex of L'' (resp. M) can access a sprout in $\mathcal{D}om(\xi_{L''})$ (resp. $\mathcal{D}om(\xi_M)$) by definition of cyclic extensions, hence a root in M (resp. L''). Since these drags are finite, $M \otimes_\xi L''$ must be a collection of cycles (up to its sprouts), hence is a cyclic drag. Therefore, $M \otimes_\xi L = L' \otimes_{\zeta'} ((L'' \otimes_\xi M) \otimes_{\zeta'''} L''')$, where ζ', ζ''' are directed and $L'' \otimes_\xi M$ is a cyclic drag.

Since L and R have the same number of roots and $\mathcal{V}ar(R) \subseteq \mathcal{V}ar(L)$ by the definition of a graph rule, $M\xi$ is also a cyclic extension for R; hence, we get extensions $R'\sigma'$, $R'''\sigma'''$, and $R''\xi$ such that $M \otimes_\xi R = R' \otimes_{\sigma'} ((R'' \otimes_\xi M) \otimes_{\sigma'''} R''')$, where σ', σ''' are directed and $R'' \otimes_\xi M$ is a cyclic drag.

Thirdly, we show by induction on $|L \otimes_\xi M| + |R \otimes_\xi R|$ that each formula

$$M \otimes_{\xi_i} L \succ M \otimes_{\xi_i} R$$

is logically equivalent to a purely universal first-order formula over the head-order predicate, and which is identical for all $M \in \{M_i^j\}_j$. This formula is obtained by unfolding the definition of GPO according to the heads of the compared drags. There are four different cases depending on the location of the head on both sides of the compared drags. Note that $M \otimes_\xi L''$ and $M \otimes_\xi R''$ cannot be empty, since switchboards are well-behaved with respect to both L and R.

1. Assume first that L', R' are nonempty, hence $L' = \widehat{L'} \otimes_\delta \nabla L'$ and $R' = \widehat{R'} \otimes_\eta \nabla R'$, where δ, η are directed. By the induction hypothesis, we get the ordering constraints C_1, C_2, and C_3 for, respectively, $\nabla L' \otimes_{\zeta'} ((M \otimes_\xi L'') \otimes_{\zeta'''} L''') \succeq R' \otimes_{\sigma'} ((M \otimes_\xi R'') \otimes_{\sigma'''} R''')$, $\nabla L' \otimes_{\zeta'} ((M \otimes_\xi L'') \otimes_{\zeta'''} L''') \succ \nabla R' \otimes_\sigma ((M \otimes_\xi R'') \otimes_{\sigma'''} R''')$, and $L' \otimes_\zeta ((M \otimes_\xi L'') \otimes_{\zeta'''} L''') \succ \nabla R' \otimes_\sigma ((M \otimes_\xi R'') \otimes_{\sigma'''} R''')$.

   We therefore get the equivalent formula $C_1 \vee (\widehat{L'} \doteq \widehat{R'} \wedge C_2) \vee (\widehat{L'} \succ \widehat{R'} \wedge C_3)$.

2. If R' is empty and L' is nonempty, then the left-hand side head originates from L' while the right-hand side head originates from $M \otimes_\xi R''$. By the induction hypothesis, we get the ordering constraints C_4 for $\nabla L' \otimes_\zeta ((M \otimes_\xi L'') \otimes_{\zeta'''} L''') \succeq (M \otimes_\xi R'') \otimes_{\sigma'''} R'''$; C_5 for $\nabla L' \otimes_\zeta ((M \otimes_\xi L'') \otimes_{\zeta'''} L''') \succ R'''$ (noting that $\nabla(M \otimes_\xi R'') = \varnothing$ since $M \otimes_\xi R''$ is cyclic); and C_6 for $L' \otimes_\zeta ((M \otimes_\xi L'') \otimes_{\zeta'''} L''') \succ R'''$.

   We therefore get the equivalent formula $C_4 \vee (\widehat{L'} \doteq M \otimes_\xi R'') \wedge C_5 \wedge C_6 \vee (\widehat{L'} \succ M \otimes_\xi R'') \wedge C_6$. Pulling out disjunctions, this reduces to C_4 since $\widehat{L'} \doteq M \otimes_\xi R''$ and $\widehat{L'} \succ M \otimes_\xi R''$ both reduce to False.

3. $L' = R' = \varnothing$. By the induction hypothesis, we get the formula C_7 for $L''' \succ (R'' \otimes_\xi M) \otimes'''_\sigma R'''$ (which must be false); C_8 for $L''' \succeq R'''$; and C_9 for $(L'' \otimes_\xi M) \otimes_{\zeta'''} L''' \succ R'''$. Since the head order is delible, $M \otimes_\xi L'' \doteq M \otimes_\xi R''$ and $M \otimes_\xi L'' \succ M \otimes_\xi R''$ reduce to $L'' \doteq R''$ and $L'' \succ R''$, respectively. We finally get the formula: $L'' \doteq R'' \wedge C_8 \wedge C_9 \vee L'' \succ R'' \wedge C_9$.

4. L' is empty and R' is nonempty. This case may occur if $\mathcal{V}ar(R) \subsetneq \mathcal{V}ar(L)$ and reduces to $L'' \succ R'' \wedge C_{10}$, the formula obtained from $(L'' \otimes_\xi M) \otimes_{\zeta'''} L''' \succ (R'' \otimes_\xi M) \otimes_{\sigma'''} R'''$. □

As a direct consequence of Lemma 37, we get a major property of GPO:

Theorem 38 (Decidability)**.** *It is decidable whether a finite drag rewriting system $\mathcal{G}$ terminates with GPO provided the rewrite head order is delible and solvable.*

Demanding finiteness of the rewriting system to be checked for termination is usually not a big restriction, but it does become one in the presence of sharing. Since all roots to be connected need to appear in the list of roots of the left-hand (and right-hand) side, using a rule with a different number of roots requires a copy of that rule with those duplicated roots appearing in the list of roots. If sharing can be arbitrary, then the rewrite system becomes infinite. If we restrict drags to have a bounded degree, then decidability is retained. More generally, we conjecture decidability in case all rules originate from a finite schema.

6 Ordering Heads

In this section, we define several different head orders. The first will be total but indelible. The second will be delible, but non-total. Both are defined by first interpreting a drag head in terms of the function symbols labeling its vertices and, secondly, by deriving an order on these interpretations from an order on the set of function symbols.

Precedence. Let $>$ be a well-ordering of symbols in $\Sigma \cup \Xi \cup \{\Box\}$, called the *precedence*.

Total head order. Represent a drag head as a list of function symbols labeling its accessible vertices in some order. Specifically, the *interpretation* of the head D of a clean drag U, written $[\![D]\!]$, is a list of elements in $\Sigma \cup \Xi \cup \{\Box\} \cup \mathbb{N}$ obtained by traversing D in a depth-first search manner, starting from the list $\mathcal{R}(D)$ of its roots, and inserting the label l of the visited vertex v within the partially-constructed linear sequence, except: if $l \in \mathcal{V}ar(D) \backslash \mathcal{V}ar(U)$, then l is replaced by $\Box$; if v has already been visited at ordinal position $n \in \mathbb{N}$ for the first time, then l is replaced by n. (Note that these two exceptions are incompatible.)

Example 39. *Consider the drags* $D = r{:}f(p{:}g(g(\to r)), \to p, x)$ *and* $D' = r{:}f(g(q{:}g(\to r)), \to q, a)$*, with an obvious notation for pointers and* r *the sole root. We have* $[\![\widehat{D}]\!] = [\![D]\!] = f\ g\ g\ 1\ x$ *and* $[\![\widehat{D'}]\!] = f\ g\ g\ 2\ \Box$*. Were* p *also a root of* D*, then* $[\![D]\!] = f\ g\ g\ 1\ x\ 1$*.*

We can now define our first head order on drag heads:

Definition 40. $D \geq D'$ *iff* $\langle \mathcal{V}ar(D), |D|, [\![D]\!] \rangle\ (\supseteq, \geq_{\mathbb{N}}, \geq_{lex})_{lex}\ \langle \mathcal{V}ar(D'), |D'|, [\![D']\!] \rangle$*, where interpretations are compared lexicographically in the precedence.*

Proposition 41. $\geq$ *of Definition 40 is a head order that is total on (non-isomorphic) heads.*

Because cyclic extensions may change the order in which vertices are traversed in a depth first search manner, this head order isn't delible. It is not a rewrite order either, since ordering the variables makes it non-stable: totality and well-foundedness of the generated GPO are the only required properties to carry out Gröbner bases computations with operadic expressions.

Rewrite head order. Represent a drag head as a multiset of function symbols labeling its accessible vertices. This time define the *interpretation* $[\![D]\!]$ of head $D = \widehat{U}$ to be the multiset $L(\mathcal{A}cc(D))$ of symbols in $\Sigma \cup \Xi$ that label the vertices of $\widehat{U}$, ignoring variables not in $\mathcal{V}ar(U)$.

To compare two drags, we define:

Definition 42. $\widehat{U} \geq \widehat{U'}$ *iff* $[\![\widehat{U}]\!] \geq_{mul} [\![\widehat{U'}]\!]$*.*

Proposition 43. $\geq$ *of Definition 42 is a delible, solvable rewrite head order.*

Proof. $\geq$ is an order whose strict part is well-founded and compatible. Furthermore, the order is stable, delible, and solvable, these being well-known properties of multiset comparisons. □

For example, we can now show termination of the rules of Figure 2. Shrinking is easy; an empty precedence suffices. Breaking requires the precedence $f \succ a$. The user can verify that GPO decreases for both rewrites given in the example.

By the same token, one can compare the multiset $X_{U^\sharp}$ of edges in the heads, with each edge treated as an ordered pair of function symbols. Consider a rule $f(f(x)) \to f(g(f(x)))$, the precedence $f > g$, and a head ordering that compares the multisets of edges. Applying the rule in a cycle results in a decrease, since $\{f{-}f, f{-}x\} >_{mul} \{f{-}g, g{-}f, f{-}x\}$, while applying it elsewhere results in an increase, adding edge $g{-}f$.

Many other variants of head order seem possible.

In the case of trees, RPO has an ordinality that makes it inherently more powerful than orders based on simple counting arguments, or even those developed in [2]. Of course, GPO has an ordinality that is at least at high as RPO. It therefore inherits the same power, as shown next with an example adapting the famous hydra game to drags.

Example 44. *The* graphical hydra game *is a simplified version of the famous hydra game but based on drags. Chopping off a head of the hydra whose power is n engenders a tentacle ending up in a head of power $n-1$ or in a ring all of whose heads have power $n-1$. The generated tentacle or ring may have an arbitrary length. A hydra will therefore be a binary tree all of whose branches end in either a head or a tentacle ending in a head or a ring of hydras. Here is the vocabulary, using dependent types:*

$$\begin{array}{rcl} \mathit{hydra} & : & \mathit{nat} \to * \\ \mathit{head} & : & \mathit{nat} \to \mathit{hydra} \\ \mathit{tent} & : & \mathit{hydra} \to \mathit{hydra} \\ \mathit{nil} & : & \mathit{hydra} \\ \mathit{append} & : & \mathit{hydra} \times \mathit{hydra} \to \mathit{hydra} \end{array}$$

The type nat of natural numbers is assumed, with constructors 0 and s. It represents the power of a given head, hence of a hydra. The branching constructor append will also be used to make cycles by using a back-pointer denoted by SELF*, which stands for an edge going back to the root of the algebraic expression in which it occurs. When occurring in an empty expression,* SELF *denotes an empty cyclic structure.*

To build a tentacle or a ring, we need additional symbols that will later be defined by rewrite rules:

$$\begin{array}{rcl} T & : & \mathit{nat} \times \mathit{nat} \to \mathit{hydra} \\ R & : & \mathit{nat} \times \mathit{nat} \to \mathit{hydra} \end{array}$$

In these declarations, the first argument stands for the power of the hydra heads, while the second is an auxiliary argument used to control the size of the tentacles or rings that are built.

Since we do not have dependent types in our framework, we used instead the infinite signature

$$\Sigma \stackrel{\text{def}}{=} \{\mathit{head}_n, T_n, R_n, \mathit{append}, \mathit{nil}, \mathit{tent}, s, 0\}$$

in which the index n is always equal to the first argument of the indexed symbol, and represents the allowed power for a head.

Here are the rewrite rules of the graphical hydra game:

$$\begin{array}{lcrcl}
\mathit{chop_head_make_head} & : & \mathit{head}_{s(n)}(s(n)) & \to & \mathit{head}_n(n) \\
\mathit{chop_head_make_tentacle} & : & \mathit{head}_{s(n)}(s(n)) & \to & T_n(n,m) \\
\mathit{chop_head_make_ring} & : & \mathit{head}_{s(n)}(s(n)) & \to & \mathit{append}(R_n(n,m), \textsc{self}) \\
\mathit{chop_tree_right} & : & \mathit{append}(x,y) & \to & x \\
\mathit{chop_tree_left} & : & \mathit{append}(x,y) & \to & y \\
\mathit{chop_tentacle} & : & \mathit{tent}(x) & \to & x \\
\mathit{grow_tentacle} & : & T_n(n,s(y)) & \to & \mathit{tent}(T_n(n,y)) \\
\mathit{empty_tentacle} & : & T_n(n,y) & \to & \mathit{head}_n(n) \\
\mathit{grow_cycle} & : & R_n(n,s(m)) & \to & \mathit{append}(\mathit{head}_n(n), R_n(n,m)) \\
\mathit{empty_cycle} & : & R_n(n,m) & \to & \mathit{nil}
\end{array}$$

The two rules chop_head_make_tentacle and chop_head_make_ring should be understood as rule schemas, m standing for an arbitrary natural number of the form $s^m(0)$. The same holds for the rule in which the natural number n occurs as an index of some function symbol. So, the only variables in these rules are x and y.

Note that the rule chop_tree_right may remove (part of) cycles. For example, it rewrites the hydra expression $\mathit{append}(a, \mathit{append}(b, \mathit{append}(c, \textsc{self})))$, where a, b, c are hydra expressions, into the hydra expression $\mathit{append}(a, b)$ when applied to the middle append vertex.

We now show termination of these drag rewrite rules by using the precedence

$$\mathit{head}_{s(n)} \succcurlyeq \{T_n, R_n\} \succcurlyeq \{\mathit{head}_n\} \succcurlyeq \{\mathit{append}, \mathit{nil}, \mathit{tent}, s, 0\}$$

which explains why we need to index some function symbols by their first argument in order to carry out the termination proof.

We first compare the rules themselves (that is, we use a trivial cyclic extension). All comparisons except the third use MPO (RPO with a multiset interpretation of all function symbols), and go through without difficulty by using the cases of the definition. For the third rule, the heads of the left-hand and right-hand side drags are interpreted, respectively, by the multisets $\{\mathit{head}_{s(n)}\}$ and $\{\mathit{append}\}$, the first being strictly bigger than the second, which now yields the recursive comparison $\mathit{head}_{s(n)}(s(n)) \succ R_n(n,n)$, which latter succeeds by a usual MPO comparison.

We are now left with checking the cyclic extensions of the rules that contain variables, that is, rules chop_tree_right, chop_tree_left, chop_tentacle, grow_tentacle and _tentacle. Since the first three are projections, their cyclic extensions are ordered like the rules themselves (This is an easy lemma to prove.) For the last, the check is trivial since the variable y does not occur on the right. We are left with the cyclic extension $\langle M_z^1, \{y \mapsto 1, z \mapsto 1\}\rangle$ of the rule grow_tentacle. This yields the formula $\forall M_z^1.T_n(n,s(y)) \otimes_{y\mapsto 1, z\mapsto 1} M_z^1 \succ \mathit{tent}(T_n(n,y)) \otimes_{y\mapsto 1, z\mapsto 1} M_z^1$ which reduces to (omitting the quantification) $\{T_n, s\} \cup [\![M]\!] \succ_{mul} \{\mathit{tent}, T_n\} \cup [\![M]\!] \wedge T_n(n,s(y)) \otimes_{y\mapsto 1, z\mapsto 1} M_z^1 \succ \{n, y\}$, where n is a given natural number. This computation succeeds therefore easily.

This terminates the termination proof of the graphical hydra game.

Note that drags make it easy to complicate the hydra game by using several chop rules at the same time: it suffices to write a left-hand side with several different roots, such as, for example

$$\mathit{append}(x,y) \quad \mathit{tent}(z) \;\to\; x \quad z$$

Proving termination of this more complicated hydra game would of course generate more comparisons, in particular, more cyclic extensions to be checked, but would not be any more difficult.

7 Conclusion

We have shown that reduction orderings defined for terms scale to the framework of drags. Using them to show termination of a rewrite system, however, necessitates comparing all (infinitely many) cyclic extensions of a rewrite rule.

Furthermore, we have designed an ordering, GPO, that can be used for comparing drags. Such an order was known for dags and jungles [9], but not for this very general kind of graphs with cycles, which raises new kinds of conceptual problems.

GPO has multiple incarnations depending on the properties of its main ingredient, the head order. The first incarnation yields a total order on drags, which can then be extended to polynomials over drags, such as operadic expressions. The second yields a reduction order over drags for which the infinite test is decidable under a natural assumption on its head-order component. This property should be investigated in greater detail to see whether it applies to other kinds of reduction orderings, in particular, those that are interpretation based.

Whether decidability of the universal fragment of head constraints is necessary to check termination of a finite set of graph rules with GPO is by no means clear. It would be interesting to know whether, under appropriate monotonicity assumptions on the head order, the infinite termination test can be reduced to a finite one, restricted to a well-chosen, statically defined, set of cyclic extensions of the rules. We believe that this question is related to our conjecture that decidability is preserved for infinite rewrite systems generated from a finite schema.

Note also that we have not found an order on cyclic drags that is at the same time total and delible (or total and monotonic); perhaps no such order exists.

One possible generalization of our framework is the use of function symbols with variable arity. This can of course be encoded in the present drag framework to the price of having infinite signatures and rewrite rules. As a result, this would increase the ordinality of the previous hydra game, if cutting a head of the hydra results in an arbitrary multiset of new heads, all having a strictly smaller power.

For a more substantial one, note that GPO reuses Rubio's idea [10] of building a desired congruence over expressions via a definition of subterm which is invariant under the congruence: associativity and commutativity of some function symbols labeling trees in his case, and isomorphism of drags in our case. We believe this technique is the right way to build a congruence on terms in a recursive path order. We plan to use it again to allow some vertices in a graph to be labeled by associative-commutative symbols, which is more general than polynomials over finite graphs.

There is another very useful way to generalize RPO that should lead to an equally useful generalization of GPO, namely, replacing the comparison of the head symbols f and g in an RPO comparison $s = f(\overline{s}) \succ g(\overline{t}) = t$ by the comparison, under an appropriate well-founded ordering, of s and t themselves. This is the usual way to build interpretation techniques into RPO. This extension of RPO indeed yields term orderings that combine the flexibility of interpretation-based orderings with the strength of RPO. We could do the same with GPO, using as the head order a suitable comparison of the entire drags to be compared. We have not explored this path.

References

[1] Bonfante, G., Guillaume, B.: Non-simplifying graph rewriting termination. In: Echahed, R., Plump, D. (eds.): Proceedings of the 7th International Workshop on Computing with Terms and Graphs (TERMGRAPH), Rome, Italy (2013) 4–16

[2] Bruggink, H.J.S., König, B., Nolte, D., Zantema, H.: Proving termination of graph transformation systems using weighted type graphs over semirings. In: Parisi-Presicce, F., Westfechtel, B. (eds.): Proceedings of the 8th International Conference on Graph Transformation (ICGT), held as part of STAF 2015. Lecture Notes in Computer Science, Vol. 9151., L'Aquila, Italy, Springer (2015) 52–68

[3] Dershowitz, N.: Termination of rewriting. J. Symb. Comput. **3** (1987) 69–115

[4] Dershowitz, N., Jouannaud, J.P.: Drags: A simple algebraic framework for graph rewriting. In: Fernandez, M., Mackie, I. (eds.): Proceedings of the 7th International Workshop on Computing with Terms and Graphs (TERMGRAPH), Oxford, England (2018)

[5] Dershowitz, N., Jouannaud, J.P.: Drags: An algebraic framework for graph rewriting. To appear in *TCS* and available at https://hal.inria.fr/hal-01853138 (2018)

[6] Dotsenko, V., Bremner, M.: Algebraic Operads: An Algorithmic Companion. CRC Press (2016)

[7] Habel, A., Kreowski, H., Plump, D.: Jungle evaluation. Fundam. Inform. **15** (1991) 37–60

[8] Jouannaud, J.P., Rubio, A.: Polymorphic higher-order recursive path orderings. J. ACM **54** (2007)

[9] Plump, D.: Simplification orders for term graph rewriting. In: Prívara, I., Ruzicka, P. (eds.): Proceedings of the 22nd International Symposium on Mathematical Foundations of Computer Science (MFCS'97), Bratislava, Slovakia. Lecture Notes in Computer Science, Vol. 1295. Springer (1997) 458–467

[10] Rubio, A.: A fully syntactic AC-RPO. Inf. Comput. **178** (2002) 515–533