WELL-FOUNDED PATH ORDERINGS FOR DRAGS

NACHUM DERSHOWITZ AND JEAN-PIERRE JOUANNAUD

Abstract. We are interested in the generalization of term rewriting techniques to what we call *drags*, viz. finite, directed, ordered multigraphs. Each vertex is labeled by a function symbol, the arity of which governs its degree in the graph. These graphs are viewed here as terms with sharing and back-arrows. We develop algebraic properties of drags and then define a *graph path ordering* on them, which is inspired by the recursive path ordering on terms, and which enjoys many of the properties that have made the latter popular, in particular, well-foundedness and – under certain circumstances – monotonicity. Our graph ordering provides an initial building block for rewriting with such 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 [5], 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 first-order terms with both sharing and back-arrows, that is, in rooted graphs (actually, multigraphs), each vertex of which is labeled by a function symbol whose arity governs the number of vertices it relates to in the graph via an outgoing edge. These graphs, qua expressions, can be composed to form monomials. In turn, these monomials can be added to form polynomials, addition being associative and commutative. These finite graphs and their polynomials are seen as expressions of an algebraic structure that we refer to generically as an *operad*. Rewriting terms of an operad is very similar to rewriting terms of an algebra. The same questions arise: What rewriting relation do we use? Is there an efficient pattern matching algorithm? Is rewriting terminating? Is it confluent?

Date: February 6, 2018.

Operadic expressions are structured: graphs and their polynomials. This somewhat eases the case of operadic expressions since, once a total order is obtained for graphs, it can be easily extended to one for polynomials by standard techniques originating from Gröbner bases. We therefore concentrate our work on graphs.

Rewriting with graphs enjoys a long history in computer science, graphs being used to represent not only data structures, but also program structures, and even computational models. The above questions have therefore been addressed by the rewriting community since the mid-eighties. Termination and confluence techniques have been elaborated for various generalizations of trees, such as rational trees, directed acyclic graphs, lambda terms and lambda graphs. But the design of tools for rewriting arbitrary graphs has made little progress beyond various definitions and the study of the particular cases we just mentioned.

This paper describes, first of all, the design of a general class of graphs – actually, multigraphs – that has good structural properties with respect to graph rewriting. We define operations for their composition and decomposition. Secondly, the paper examines appropriate well-founded orderings that can be used to show termination of rewriting on those graphs. Of course, different target applications require different properties from a well-founded graph ordering: totality is essential for operaders, while monotonicity with respect to the graph structure is important for graph rewriters.

A popular ordering for ordinary terms is the recursive path ordering (RPO) [4], which is recursively generated from an order on the set of function symbols, called the "precedence". One reason for its utility is that it enjoys all the good properties one may wish for: it is compatible with term structure; it is well-founded when the precedence is; it is increasing when the precedence increases; and – last but not least – its behavior is intuitive and can be easily and efficiently implemented. One main aspect of its definition is that comparing two terms with the same head symbol reduces to a comparison of their respective subterms via a lexicographic or multiset extension of the order on their subterms organized as lists or multisets, respectively. The type of extension assigned to a head symbol is referred to as its *status*. When statuses are lexicographic and the precedence total, RPO itself becomes total.

We propose a generalization of RPO to ordered, labeled, rooted graphs. The ordering we end up with, *GPO*, has the very same definition as RPO, 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. A key property of our decomposition is that isomorphic multigraphs have isomorphic heads and tails. This alone ensures that the order is compatible with multigraph isomorphism. GPO has many of the properties that are important for its various users. It is well-founded, total on graph expressions up to isomorphism, and has some good monotonicity properties.

Graphs and their decomposition are studied in Section 2, GPO in Section 3, its properties in Section 4. Ordering drag heads comes in Section 5. Vexing open problems are described in the discussion.

2. 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. In the present work, 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 (we have no associative-commutative symbols here), 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.

The successors of a vertex labeled f in a drag are implicitly interpreted as the arguments of the function symbol f. The graph describes therefore the inverse of the computational flow. Some might prefer the converse choice. Note also that this class of graphs is the most general one can think of to describe computations in the absence of binding constructs. Indeed, Lafont's interaction nets [7] appear as a particular case of the drag model.

To lessen notational burden, we use the 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 list concatenation as well as set and multiset union, $\setminus$ for list, set or multiset difference, and identify a singleton list, set or multiset with its content.

2.1. **Drags.**

Definition 2.1 (Drag)**.** *A* (closed) drag *is a tuple $\langle V, R, L, X\rangle$, where V is the finite set of* vertices*; R is a finite list of vertices, called* roots*, and $R(n)$ is the nth root in the list; $L : V \to \Sigma$ is the* labeling *function, mapping vertices to labels from some vocabulary Σ; and $X : V \to V^*$ is the* successor *function, mapping each vertex $v \in V$ to a list of vertices in V whose length equals the appropriate arity $|L(v)|$. Given $b \in X(a)$, then (a, b) is an* edge *with* source *a and* target *b. We also write aXb. The transitive closure X^* of the relation X is called* accessibility*. A vertex v is said to be* accessible *if rX^*v for some root r. A drag is* clean *if all its vertices are accessible. A root r is* maximal *if $\forall r' \in R.\ r'X^*r$ implies rX^*r'.*

Definition 2.2 (Open drag)**.** *An* open drag *is a drag over the set $\Sigma \cup \Xi$, where Ξ is a set of* variables*. The vertices labeled by a function symbol in Σ are* internal*; those labeled by a variable are called* sprouts*. The set of sprouts will usually be denoted by the letter S and added at the end of the tuple when it's nonempty. An open drag is* linear *if different sprouts have different labels. It is* cyclic *if $\forall v \in V \setminus S.\ \exists r \in R.\ rX^*vX^*$.*

The component R is a list with repetitions, whereas S is a set. Allowing for repeated roots is needed for rewriting. It will sometimes be convenient to consider roots as specific incoming edges. Sprouts may be roots. Cleanness relates to garbage collection, discussed further in Section 2.4.

A cyclic drag may be reduced to a single internal vertex; hence the ground term a and the term $f(x)$ correspond to cyclic drags while the terms $f(a)$ and $f(f(x))$ do not. Terms, sequences of terms and terms with sharing are particular drags. Nonlinear drags play an essential rôle for sharing and unsharing subterms in terms.

Given a drag D, we use the following notations: $\mathcal{V}er(D)$ for its set of vertices; X_B for its successor function; $\mathcal{A}cc(V)$ for its set of accessible vertices; $\mathcal{R}(D)$ for its set of roots; $\mathcal{R}(D)_{max}$ for the subset of *maximal* roots; $\mathcal{V}ar(D)$ for the set of variables labeling its sprouts; $D^{\overline{r}}_{\overline{s}}$ to indicate its roots and sprouts; and $D{\downarrow}$ for the clean drag obtained by removing its inaccessible vertices and related edges. labelings extend to lists, sets and multisets of vertices as expected. We often identify sprouts with their variable labels.

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:

Definition 2.3 (Drag isomorphism)**.** *Given two open drags* $D = \langle V, R, L, X, S\rangle$ *and* $D' = \langle V', R', L', X', S'\rangle$ *whose sprouts are identified with their labels, an* isomorphism *from* D *to* D' *is a one-to-one mapping* $o : \mathcal{A}cc(V) \mapsto \mathcal{A}cc(V')$ *such that*

(i) o *restricts to bijections between the lists of roots and sets of accessible sprouts;*
(ii) $\forall v \in \mathcal{A}cc(V) \setminus S.\ L(v) = L'(o(v))$ *and* $X(v) = X'(o(v))$*;*

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.*

Note that inaccessible vertices are ignored, making any drag D isomorphic (with $\equiv$) to $D{\downarrow}$. Identifying the sprouts with the variables that label them saves an additional one-to-one mapping between the variables that would otherwise become necessary.

Example 2.4. *Our running example is made of seven single-rooted drags represented in Figure 1.*

The pictorial representation does not tell us precisely which argument of a function symbol comes first. In our convention, the leftmost one in the figure comes first. Another convention tells us the order of arguments, by sweeping

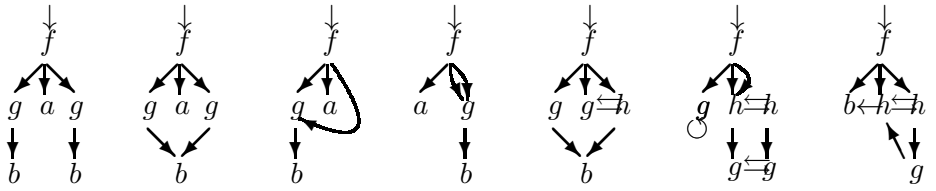

FIGURE 1. Seven simple drags whose vertices are identified with the extremities of their edges.

the plane counterclockwise from the first. Finally, an incoming down-arrow indicates a root.

Ariola and Klop's wording [1], horizontal/vertical sharing, *is intuitive. Let* $\Sigma = \{a, b, f, g, h\}$ *with* $|f| = 3$, $|h| = 2$, $|g| = 1$ *and* $|a| = |b| = 0$. *We first consider the term* $f(g(b), a, g(b))$ *in which* b *and* $g(b)$ *can be shared. Among all possible cases, 4 are represented. The 5th has both horizontal and vertical sharing. The 6th has three cycles: an independent leftmost loop and a horizontal cycle sharing another horizontal cycle below. The last has a single big cycle, which is both horizontal and vertical including an inside cycle.*

2.2. **Drag Algebra.** We equip drags with a composition operator, which, given two drags, connects the sprouts of each drag with the roots of the other. As usual, we 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 subset A of its domain to subset B of its image, omitting $\to B$ when irrelevant. First, comes the connection device:

Definition 2.5 (Switchboard)**.** *Let* $D = \langle V, R, L, X, S\rangle$ *and* $D' = \langle V', R', L', X', S'\rangle$ *be open drags. A* switchboard ξ *for* (D, D') *is 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* $D_{|R'(\xi_D(s))} \simeq D_{|R'(\xi_D(t))}$*;*
(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* $D'_{|R(\xi_{D'}(s))} \simeq D'_{|R(\xi_{D'}(t))}$*.*

Conditions (i,ii) are always satisfied for linear drags, or for nonlinear drags whose switchboard, called *linear*, is defined for sprouts whose variables are all different. When this is not the case, checking that a particular ξ is a switchboard involves graph isomorphism, which is polynomial in our case, since we have ordained fixed arities. (Were one to allow varyadic vertices, isomorphism would be pseudo-polynomial [2].) A simple effective approximation here is to require instead that $R'(\xi_D(s)) = R'(\xi_D(t))$ for (i) and similarly $R(\xi_{D'}(s)) = R(\xi_{D'}(t))$ for (ii). This approximation is not expressive enough for the case of trees, but turns out to fit shared trees, for which

isomorphism is computable in linear time. In the sequel, we assume this latter condition for simplicity.

In practice, we shall usually enforce $\mathcal{V}ar(D) \cap \mathcal{V}ar(D') = \varnothing$ so as to avoid confusion – in particular when writing a switchboard as a union of mapping, but this is by no means necessary since switchboards are pairs of mappings operating on $\mathcal{V}ar(D)$ and $\mathcal{V}ar(D')$ independently, not a single mapping operating on $\mathcal{V}ar(D) \cup \mathcal{V}ar(D')$.

Since $\mathcal{I}m(\xi_D)$ is a list of numbers, selecting roots of D, we denote by $R \setminus \mathcal{I}m(\xi_D)$ the list of roots of D obtained from R by filtering out those selected by $\mathcal{I}m(\xi_D)$. Note that injectivity of ξ and the fact that roots are lists with repetitions go along together. One could think of using sets for roots, in which case ξ would not be injective. This alternative, however, does not fit well with rewriting for which it will be essential to control precisely both the number and order of roots of left- and right-hand sides of rules.

Three particular kinds of switchboards play important rôles:

Definition 2.6 (Categorization)**.** *A switchboard ξ for (D, D') is*

- directed *if one of ξ_D and $\xi_{D'}$ has an empty domain;*
- preserving *if ξ_D or $\xi_{D'}$ is preserving, where ξ_D (resp., $\xi_{D'}$) is* preserving *if $\forall x \in \mathcal{V}ar(D)$, $R'(\xi_D(x))$ is a nonmaximal root of D' (resp., $x' \in \mathcal{V}ar(D'), R(\xi_{D'}(x')), D$);*
- rewriting *if ξ_D is linear and surjective and $\xi_{D'}$ is total.*

Directed switchboards correspond to the tree case, with all connections from one drag to the other. Preserving switchboards are more general, allowing connections from, D', say, to nonmaximal roots of D, hence to the "tail" of D (to be defined later). Rewriting switchboards allow one to "encompass" drag D' in drag D.

Definition 2.7 (Composition)**.** *Let $D = \langle V, R, L, X, S\rangle$ and $D' = \langle V', R', L', X', S'\rangle$ be open drags, and ξ be a switchboard for (D, D') such that $(V \setminus \mathcal{D}om(\xi_D)) \cap (V' \setminus \mathcal{D}om(\xi_{D'})) = \varnothing$. Their* composition *is the drag $D \otimes_\xi D' \stackrel{\text{def}}{=} \langle V'', R'', L'', X'', S''\rangle$, where*

(1) $V'' = (V \setminus \mathcal{D}om(\xi_D)) \cup (V' \setminus \mathcal{D}om(\xi_{D'}))$;
(2) $R'' = (R \setminus \mathcal{I}m(\xi_{D'}) \setminus \mathcal{D}om(\xi_D)) \cup R(\xi_{D'}((R' \setminus \xi_D(S)) \cap S')) \cup (R' \setminus \mathcal{I}m(\xi_D) \setminus \mathcal{D}om(\xi_{D'})) \cup R'(\xi_D((R \setminus \xi_{D'}(S')) \cap S))$;
(3) $L''(v \in V \setminus \mathcal{D}om(\xi_D)) = L(v)$ and $L''(v \in V' \setminus \mathcal{D}om(\xi_{D'})) = L'(v)$;
(4) $X''(v \in V \setminus S) =$ if $X(v) \notin \mathcal{D}om(\xi_D)$ then $X(v)$; otherwise $R(\xi_D(v))$ and $X''(v' \in V' \setminus S') =$ if $X'(v') \notin \mathcal{D}om(\xi_{D'})$ then $X'(v')$; otherwise $R'(\xi_{D'}(v'))$;
(5) $S'' = (S \setminus \mathcal{D}om(\xi_D)) \cup (S' \setminus \mathcal{D}om(\xi_{D'}))$.

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 vertices referred to in the switchboard images. This necessitates renaming (via ξ) the targets of edges that had pointed to the sprouts. The only complication is the calculation of the list of roots R'' of the resulting drag D'', since nodes may inherit root status from the sprouts with which they

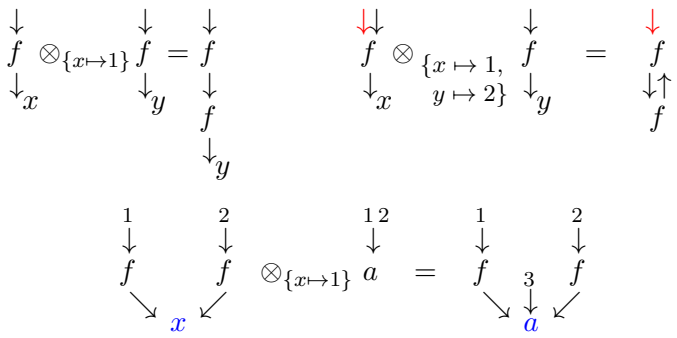

FIGURE 2. Unidirectional, cyclic and root-transfer compositions.

are merged. One also needs to worry about the case where multiple sprouts are merged, when the switchboards map sprout to rooted-sprout to rooted-sprout, possibly in a cycle. In such cases, all need to be renamed to the same one.

Note that if ξ is a rewriting switchboard, then *all* roots and sprouts of D' disappear in the composed drag. Otherwise, the symmetry of the definition is broken by choosing the roots originating from D to come first.

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.

Example 2.8. *We show in Figure 2 three examples of compositions, the first two with similar drags. The first composition uses a directed switchboard, while the second uses a rewriting switchboard which induces a cycle. Variables x, y label the sprouts. 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 roots which are also sprouts behave in the composition.*

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 a cyclic permutation of their roots.

Lemma 2.9 (Commutativity)**.** *Composition of drags is quasi-commutative, and becomes commutative for switchboards whose context or substitution is surjective, e.g. rewriting switchboards.*

Commutativity shows that there is no big difference between the context and the substitution of a switchboard in a composition $D \otimes_\xi D'$, since they simply exchange each other when exchanging the input drags. In the sequel, by a preserving or directed switchboard, we shall always mean a switchboard whose context is preserving or directed, respectively.

Let us call *identity* a linear open drag whose all vertices are its sprouts and set of edges is empty. We denote it by 1^Y_Z, where $Y \subseteq Z^*$ is its list of

roots. We use $\varnothing$ for the drag $1^{\varnothing}_{Z}$, which is isomorphic to $1^{\varnothing}_{\varnothing}$, and is called the empty drag for that reason.

Lemma 2.10 (Neutral)**.** *Let D be an open drag, $X \subseteq \mathcal{V}ar(D)$, and ι the directed identity switchboard for $(D, 1^{X}_{X})$. Then $D \otimes_{\iota} 1^{X}_{X} = D$.*

In particular, this property holds when $X = \varnothing$.

Definition 2.11 (Compatibility)**.** *Two switchboards ξ and ζ for the respective pairs of drags (U, V) and (V, W) are* compatible *if (i) $\mathcal{D}om(\xi_V) \cap \mathcal{D}om(\zeta_V) = \varnothing$ and (ii) $\mathcal{I}m(\xi_U) \cap \mathcal{I}m(\zeta_W) = \varnothing$.*

Lemma 2.12 (Associativity)**.** *Let U, V, W be three drags, and ξ, ζ be compatible switchboards for (U, V) and (V, W) respectively. Then, $(U \otimes_{\xi} V) \otimes_{\zeta} W = U \otimes_{\xi'} (V \otimes_{\zeta'} W)$.*

Proof. Compatibility ensures that ξ and ζ are switchboards for $(U, V \otimes_{\zeta} W)$ and $(U \otimes_{\xi} V, W)$, by eliminating the possibility that a sprout or root is used twice. Both sides of the associativity equation are thus defined. The equality itself results from easy checking. □

2.3. **Drag Structure.** Next, we investigate ways of decomposing a drag. We first give a general construction which splits a drag into a *subdrag* generated by some set of vertices, and the rest of the drag, its *antecedent*.

Definition 2.13 (Subdrag)**.** *Given a drag $U = (V, R, L, X, S)$, 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 (V', R', L, X, S') 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 $(R \cap V') \cup X(V \setminus V')$.

Restricting to an empty set of vertices, yields the empty drag $\varnothing \stackrel{\text{def}}{=} 1^{\varnothing}_{\varnothing} = (\varnothing, \varnothing, \varnothing, \varnothing, \varnothing)$.

Note that a subdrag is clean by construction. 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 really matter for now.

Given a switchboard ξ for (W, U) and a subdrag A of W, we denote by $\xi_{U \to A}$ the restriction of ξ_U whose image is in A.

Lemma 2.14. *Given a drag D and a list of vertices $W \subseteq V$, there exists a linear drag A, called* antecedent*, and a directed switchboard ξ such that $D = A \otimes_{\xi} D_{|W}$.*

Proof. A has for internal vertices those of D which do not belong to $D_{|W}$, and for roots the list of roots of D which are not in $D_{|W}$. On the other hand, its sprouts correspond to the roots of $D_{|W}$ which are not roots of D. These sprouts can be labeled by different variables so as to make the antecedent

linear. The switchboard ξ mapping these sprouts to the corresponding roots of $D_{|W}$ is directed since subdrags are closed under successor. □

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 defines implicitly an order on the roots of the subdrag which are not roots of the whole drag.

This construction has two particular cases considered in turn. The first is the case of an empty antecedent:

Lemma 2.15. *Given a drag D, $D{\downarrow}$ is the subdrag of D generated by its maximal roots.*

The second case is more interesting, it exhibits the smallest nontrivial antecedent of a drag.

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. This unique decomposition is a fundamental property of trees which expresses the fact that the set of trees equipped with the "head-operations" has an initial algebra structure.

Likewise, a drag has a head, which is intended to be the largest cycle containing the maximal roots of the drag, and one tail, possibly a list of several connected components. The cycle may be of zero length in order to have trees as a particular case. As for trees, the head will have a list of new sprouts which are in one-to-one correspondence with the roots of the subdrag. A drag can therefore be seen as a bipartite graph made of its head, its tail, and a directed switchboard specifying that correspondence.

Definition 2.16. *The* tail ∇D *of a drag $D = (V, R, L, X, S)$ is the subdrag generated by the set of vertices $V \setminus \{v \in V : vX^*R_{max}\}$.*

As an application of Lemma 2.14, we get:

Lemma 2.17. *Given a drag D, there exists a linear drag $\widehat{D}$, called* head *of D, and a directed switchboard ξ such that $D = \widehat{D} \otimes_\xi \nabla D$.*

The head of a drag is therefore the antecedent of its tail.

Note that ξ is a bijection between the sprouts of the head and the roots of the tail that were not already roots of the drag D. In the sequel, we assume that the sprouts of the head are ordered with respect to a depth first search initialized with its list of roots. This order on the sprouts of the head induces via the switchboard ξ, an order on the roots of the associated tail which were not roots of the original drag. The tail is therefore now canonically determined.

We could have defined the head and tail of D as a pair of drags whose head contains all maximal roots, is cyclic, and its composition with the tail by a directed switchboard is the drag D itself. It would then have been

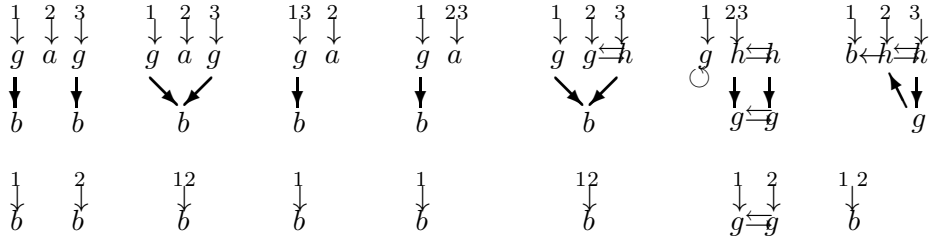

FIGURE 3. Successive subdrags of the drags of Figure 1.

necessary to show their existence. This characterization of the head of a drag is a property that we should keep in mind.

We can now show the structure theorem of open drags:

Theorem 2.1 (Structure)**.** *Isomorphic drags have isomorphic heads and tails.*

Proof. By Lemma 2.15, we restrict our attention to clean drags. Let $H \otimes_\xi T \stackrel{\text{def}}{=} \langle V, R, L, X, S\rangle = \langle V', R', L', X', S'\rangle \stackrel{\text{def}}{=} H' \otimes_{\xi'} T'$, where H and H' are heads, T and T' are open drags, $\xi_T, \xi'_{T'}$ are empty, ξ_H and $\xi'_{H'}$ are total and surjective, and o is a bijection (identifying sprouts with their labels). We build new bijections o_h and o_t that witness the isomorphisms between heads and tails respectively.

Since ξ_T is empty and ξ_H is surjective, the roots of $H \otimes_\xi T$ are those of H. Likewise, the roots of $H' \otimes_{\xi'} T'$ are those of H'. Hence o restricts to a one-to-one-mapping from the roots of H to those of H'. By property (iii) of isomorphisms, o restricts to a one-to-one mapping from the vertices of $H \otimes_\xi T$ which are vertices of H to the vertices of $H' \otimes_{\xi'} T'$ which are vertices of H', and satisfies (iii). To extend o_H to the sprouts of H and H', it suffices to notice that they are equally many by (iii). Since o satisfies (ii), so does o_h, hence $H =_{o_h} H'$.

We can now easily define o_t by removing o_h from o. The roots of T and T' correspond to the sprouts of H, H'. Since the roots of T and T' are in one-to-one correspondence with the sprouts of H and H', respectively, o_t restricts to the roots of T, T'. Also, since ξ_H and $\xi'_{H'}$ are surjective, the sprouts of T and T' are those of $H \otimes_\xi T$ and $H' \otimes_\xi T'$, respectively. Hence, o_t restricts to the sprouts of T, T'. Since o satisfies (ii), so does o_t, hence $T =_{o_t} T'$. □

Example 2.18. *The successive tails of the drags of Figure 1 are represented in Figure 3. The numbering displayed above the roots of each tail specifies the order of roots in their list of roots. These tails, represented on the first row, have all the same number of roots which is determined by their head f. The next row represents the next level of tails, which here have one or two roots, depending on their own head. The last tail in that row has two roots, a nonmaximal one in its father drag, and a new one.*

The fundamental property of a drag expressed by the previous theorem is that its decomposition into a head and tail is a faithful representation of it. This property holds true because drags are multi-rooted. Uni-rooted drags cannot represent horizontal sharing, in contrast to vertical sharing which can *sometimes* be preserved with uni-rooted drags. Moreover, the fact that tails are quasi-isomorphic does not hamper faithfulness: different orders of new roots for the tails yield different switchboards but the order of roots resulting from the composition does not depend upon those orders.

2.4. **Drag Rewriting.** The notion of composition lead in the previous section to an initial, canonical way of decomposing a drag, into its head and tail. In this section, we investigate further ways of decomposing a drag, possibly breaking its cycles, which will of course become important when rewriting drags. More precisely, can a drag D be seen as the composition of a given drag U with some context W via some switchboard ξ? In this case, we would of course say that D *matches* U, W and ξ being the *matching* context and switchboard.

The idea is that W splits into three parts: the vertices of W that are accessible from some sprout of U and can access some of its roots define a drag B; those that are accessible from some sprout of U but cannot access any of its roots define a drag C; those which are not accessible from the sprouts of U form the remaining part A. Note that $\mathcal{D}om(\xi_C) = \varnothing$.

We first need two new notions:

Definition 2.19 (Directed extension)**.** *A* directed extension *of a clean drag U is a pair (C, ξ) made of a drag $C = (V, L, R, X, S)$ and a directed switchboard ξ for (C, U).*

Particular directed extensions are identity directed extensions 1^z_z such that $\xi(x_1) = \ldots = \xi(x_n) = z$, 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 directed extensions is to modify the structure of U by introducing sharing among its sprouts. We shall see later that identity directed extensions suffice for that purpose.

We now move to the second kind of extensions:

Definition 2.20 (Cyclic extension)**.** *A* cyclic extension *of a clean drag U is a pair (B, ξ) made of a linear drag $B = (V, L, R, X, S)$ and a switchboard ξ for (B, U), called* cyclic*, such that B is generated by $\mathcal{I}m(\xi_U)$, $B \otimes_\xi U$ is a clean nonempty drag and $\forall t \in V \setminus S.\ \exists s \in \mathcal{D}om(\xi_B).\ tX^*s$.*

The conditions for being a cyclic extension impose that ξ_U is surjective on $R \setminus S$ as a set so as to generate B. The roots of the resulting drag are therefore either remaining roots of B (possibly some roots in S transfered to $B \otimes_\xi U$) or roots of U, but there must be sufficiently many of them so that the resulting drag is clean. On the other hand, ξ_B is surjective on the

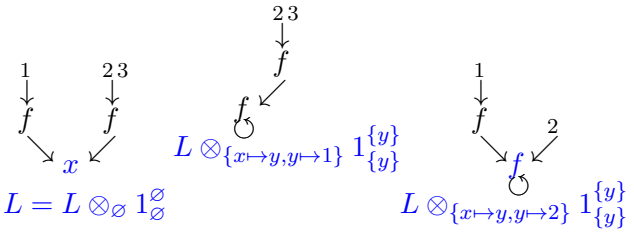

FIGURE 4. Minimal cyclic extensions.

body of U but not necessarily on U itself. Then, its body has an antecedent in U by Lemma 2.14.

Identity cyclic extensions of U are 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.

The rôle of cyclic extensions is to modify the structure of U by connecting some of its sprouts to some of its roots. Identity cyclic extensions do not suffice for that purpose.

Example 2.21. *Let L be the 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 4. 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.*

Lemma 2.22 (Decomposition)**.** *Let U, W be clean nonempty drags and ξ be a rewriting switchboard for (W, U). Let*

- *B' be the subdrag of W generated by $\mathcal{I}m(\xi_U)$ and A its antecedent such that $W = A \otimes_\zeta B'$;*
- *C be the subdrag of B' generated by the vertices of B' which cannot reach a sprout in $\mathcal{D}om(\xi_W)$ and B its antecedent such that $B' = B \otimes_\theta C$.*

Then, $W \otimes_\xi U = A \otimes_{\zeta \cup \xi_A} (B \otimes_\xi U) \otimes_{\theta \cup \xi_{U \to C}} C$, (B, ξ) is a cyclic extension of U, $(C, \xi_{U \to C})$ is a directed extension of U, the switchboards $\zeta \cup \xi_A$ and $\theta \cup \xi_{U \to C}$ are directed, and $\theta \cup \xi_B$ is total.

Since ξ is a rewriting switchboard, then it follows without saying that $\xi_A \cup \xi_B$ is surjective and $\xi_{U \to B} \cup \xi_{U \to C}$ is total.

Proof. The lemma is depicted in Figure 5. Its proof follows from Lemma 2.14 and associativity of composition for switchboards which are compatible by

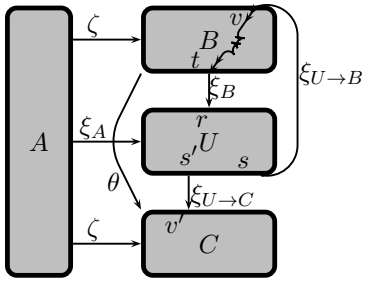

FIGURE 5. Decomposition of a pair of drags.

construction. In particular, all internal vertices of B must be reachable from one of its roots because this is true of B', and must reach a sprout belonging to $\mathcal{D}om(\xi_B)$ by the definition of C. Note also that ξ_B and θ operate on different sprouts, their union having, then, the set of sprouts of B as its domain. □

In particular, identity extensions $(1_X^Y, \iota)$ play an important rôle. They allow one to change the structure of a drag without changing its internal nodes. If the drag has a single root, it is easy to see that identity extensions are enough to predict all forms 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. Note that non-identity extensions cannot do any better if the vocabulary contains unary symbols only, as in Figure 2.21. If the vocabulary contains a symbol g with $|g| = 2$, then new cyclic extensions pop up, both with the same drag $(\{1,2,3\}, \{1\}, L(1) = g$ and $L(2) = y$ and $L(3) = z, X(1) = \{2,3\}, \{2,3\})$, and with the respective switchboards $\{x \mapsto 1, y \mapsto 1, z \mapsto 2\}$ and $\{x \mapsto 1, y \mapsto 1, z \mapsto 3\}$.

Definition 2.23 (Rules)**.** *A* graph rewrite rule *is a pair of open clean drags written $L \to R$ such that $|\mathcal{R}(L)| = |\mathcal{R}(R)|$ and $\mathcal{V}ar(R) \subseteq \mathcal{V}ar(L)$. 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.

Note that we could allow R to be unclean, and strengthen instead the second condition to be $\mathcal{V}ar(R) = \mathcal{V}ar(L)$. Adding unreachable sprouts to a clean R would then do with an isomorphic right-hand side. Such a useful trick is not possible with trees.

Definition 2.24 (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 $D' = (C \otimes_\xi R)\downarrow$ for some drag rewrite rule $L \to R \in G$, open drag C called rewriting* context *and rewriting switchboard ξ for (C, L) which is linear if L is linear.*

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. Finally, if L is linear, then ξ is a linear rewriting switchboard: no test for isomorphism is needed when pattern matching D with L.

If left-hand and right-hand sides of a drag rewrite rule have the same variables as sprouts, the computed drag generated will be clean if the starting drag is. But if there are strictly fewer variables in the right-hand side, replacement may generate inaccessible vertices, hence require garbage collection, which is therefore explicitly built in this model of computation. On the other hand, duplication is banned by the drag rewriting model, which therefore generalizes shared rewriting, but not term rewriting per se.

By commutativity of composition, the rewriting context C can also be seen as a substitution context. In the case of trees, for example, the juxtaposition of the tree-context and the substitution becomes the drag rewriting context or the drag rewriting substitution, depending on whether it appears on left or right of the composition. What is remarkable about the above definition is that there is no longer any distinction between the two, which, together, constitute the open drag rewriting context of the left-hand side. It also explains why the two components of a switchboard were named context and substitution respectively. In the sequel, we sometimes identify the rewriting (or substitution) context with the associated component of the switchboard.

Example 2.25. *Consider the cyclic composition of Figure 2, and let* $f(y') \to y'$ *be a graph rewrite rule whose left-hand and right-hand sides are, more precisely, the respective drags* $(\{1,2\}, 1, L(1) = f$ *and* $L(2) = y', X(1) = 2, \{2\})$ *and* $(\{1\}, 1, L(1) = y', , \{1\})$*. Then, the drag* $(\{1,2\}, 1, L(1) = L(2) = f, X(1) = 2$ *and* $X(2) = 1)$ *rewrites to the drag* $(\{1,2\}, 1, L(1) = f$ *and* $L(2) = x', X(1) = 2, \{2\}) \otimes_{\{x' \mapsto 1, y' \mapsto 1\}} (\{1\}, 1, L(1) = y', , \{1\}) = (\{1\}, 1, L(1) = f, X(1) = 1)$*: 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 $(\{1,2\}, 1, L(1) = L(2) = f, X(1) = 2$ *and* $X(2) = 1)$ *rewrites to the drag* $(\{1,2\}, 1, L(1) = f$ *and* $L(2) = x', X(1) = x', \{2\}) \otimes_{\{x' \mapsto 1\}} (\{1\}, 1, L(1) = a,) = (\{1,2\}, 1, L(1) = f, L(2) = a, X(1) = 2)$ *with the rule* $f(y') \mapsto a$ *whose both sides are the drags* $(\{1,2\}, 1, L(1) = f$ *and* $L(2) = y', X(1) = 2, \{2\})$ *and* $(\{1\}, 1, L(1) = a,)$*.*

These rewrites are shown in Figure 6. In both cases, the upper occurrence of f *(in blue) is part of the context, while the one below (in red) is part of*

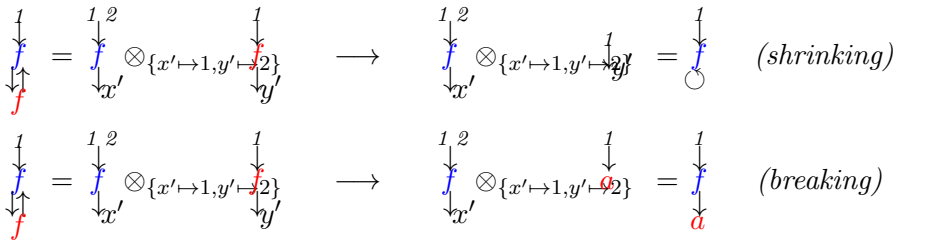

FIGURE 6. Rewriting a cycle.

the rewrite rule. Rewriting the upper occurrence of f instead of the lower is an exercise left to the reader.

Lemma 2.26. *If $U \underset{R}{\longrightarrow} V$, then $\mathcal{V}ar(V) \subseteq \mathcal{V}ar(U)$.*

Proof. The property is true of rules and preserved by rewriting. □

Lemma 2.27. *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).*

Proof. Because the lists of roots of U and V have the same length, and because the sprouts of U, V are labeled by the same variables that are here identified with the sprouts themselves, ξ_U does need any change, and since $\mathcal{V}ar(V) \subseteq \mathcal{V}ar(U)$, ξ_V is the restriction of ξ_U to the sprouts of V whose variables are in $\mathcal{V}ar(W)$. □

2.5. **Drag Orderings.** We now consider the properties that drag orderings need to satisfy for our two applications, to operads and to graph rewriting. Since the latter include the former, we will concentrate on the latter.

Definition 2.28 (Graph rewrite ordering)**.** *A* rewrite ordering *for a rewriting system $\mathcal{R}$ is an ordering $\succ$ of the set of drags which is*

(i) well-founded*: it has no infinitely decreasing chain $U_1 \succ U_2 \succ \cdots \succ U_n \succ \cdots$;*
(ii) compatible *with drag isomorphism: if $U \succ V$, $U \equiv U'$ and $V \equiv V'$, then $U' \succ V'$;*
(iii) monotonic*: if for all $L \to R \in \mathcal{R}$, then $L \succ R$ implies that $A \otimes_\xi L \succ A \otimes_\xi R$ for all drags A and directed switchboards ξ;*
(iv) directed monotonic*: if for all $L \to R \in \mathcal{R}$, there exists a finite set $\{(M, \xi)\}_{i \in I}$ of directed extensions of L, called* minimal*, such that, if $\forall i \in I.\ L \otimes_{\xi_i} M_i \succ R \otimes_{\xi_i} M_i$, then $L \otimes_\xi C \succ R \otimes_\xi C$ for all directed extensions (C, ξ) of L; if $L \to R \in \mathcal{R}$, W is an open drag and ξ is a directed rewriting switchboard for (W, L), then $W \otimes_\xi L \succ W \otimes_\xi R$;*
(v) cyclic monotonic*: if for all $L \to R \in \mathcal{R}$, there exists a finite set $\{(M, \xi)\}_{i \in I}$ of cyclic extensions of L, called* minimal*, such that, if $\forall i \in I.\ L \otimes_{\xi_i} M_i \succ R \otimes_{\xi_i} M_i$, then $L \otimes_\xi B \succ R \otimes_\xi B$ for all cyclic extensions (B, ξ) of L.*

Monotonicity is the usual property since a directed switchboard makes the context A into a usual context. Directed monotonicity corresponds to the usual stability property, but can introduce sharing as we have already seen. On the other hand, cyclic monotonicity is a new property which deals with contexts that connect a sprout of the term U to one of its roots, hence possibly creating a cycle, which explains the name. Only the minimal extension contexts, with no internal vertices, need be considered. Note that extensions of L are also extensions of R since $\mathcal{V}ar(R) \subseteq \mathcal{V}ar(L)$.

Theorem 2.2 (Termination)**.** *A graph rewrite system $\mathcal{R}$ terminates iff there exists a graph rewrite ordering $\succ$ such that for all rules $l \to R \in \mathcal{R}$, then, first, $L \otimes_\xi M \succ R \otimes_\xi M$ for all minimal directed extensions (M, ξ) of L, and second, $L \otimes_\xi M \succ R \otimes_\xi M$ for all minimal cyclic extensions (M, ξ) of L.*

Proof. Assume $U \longrightarrow_{\mathcal{R}} V$. By definition of rewriting, $U = W \otimes_\xi L$ and $V = (W \otimes_\xi R)\downarrow$. By compatibility and Lemma 2.15, it suffices to show that $W \otimes_\xi L \succ W \otimes_\xi R$ for some $L \to R \in \mathcal{R}$. By Lemma 2.22, $W \otimes_\xi L = A \otimes_\zeta ((B \otimes_\xi L) \otimes_{\theta \cup \xi_{U \to C}} C)$ for some directed switchboards ζ and θ such that (B, ξ) and (C, ξ) are cyclic and directed extensions of L, respectively.

By directed monotonicity and the assumption that $L \otimes_\iota M \succ R \otimes_\iota M$ for all minimal directed extensions (M, ι) of U, then, $L \otimes_\iota C \succ R \otimes_\iota C$. By cyclic monotonicity, the minimality assumption for cyclic extensions and associativity, $(B \otimes_\xi L) \otimes_i otaC \succ (B \otimes_\xi R) \otimes_\iota C$. By monotonicity, $A \otimes_\zeta (B \otimes_\xi L) \otimes_\theta C \succ A \otimes_\zeta (B \otimes_\xi R) \otimes_\theta C$. Hence, $W \otimes_\xi L \succ W \otimes_\xi R$ by compatibility. The result now follows from the well-foundedness of $\succ$. For the converse, the derivation relation itself defines a graph rewrite ordering for R. □

In the case of trees, left-hand sides of rules have no cyclic extensions. Further, there directed extensions cannot introduce sharing, hence the test reduces to $L \succ R$ for each rule $L \to R \in \mathcal{R}$.

In the case of shared trees, there are no cyclic extensions either, but directed extensions can now introduce sharing.

Minimal directed extensions are easy to define for the general case of drags. Assume, for example, that L has two different sprouts labeled by the variables x, y, y being possibly x itself. Composing L with the identity drag 1^z_z via the switchboard ξ such that $\xi(x) = \xi(y) = z$ will rename both x and y to z and map the two different sprouts to the root of the identity drag, thereby introducing sharing. Here, minimal directed extensions are just those identity extensions that allow us to merge different sprouts. There are of course finitely many.

Definition 2.29. *Given a drag U, and a partition of $\mathcal{V}ar(U)$ into subsets $X_1, \ldots, X_n$ we call minimal directed extension of L the pairs $(1^Y_Y, \iota)$ where $Y = \{y_1, \ldots, y_n\}$ and $\iota(x) = i$ iff $x \in X_i$.*

Minimal directed extensions are of course finitely many. Furthermore:

Lemma 2.30. *For any directed extension (C, ξ) of U, there exists a minimal directed extension (M, ι), such that $U \otimes_\xi C = (U \otimes_\iota M) \otimes_\theta N$ for some drag N and directed switchboard θ.*

Proof. Easy construction left to the reader. □

Using the lemma, it then follows that $L \otimes_\iota M \succ R \otimes_\iota M$ implies $C \otimes_\xi L \succ C \otimes_\xi R$ by monotonicity, hence showing that this set of minimal directed extensions has the required property.

3. Graph Path Ordering

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 3.1 (Head order)**.** *A* head order *for a drag rewrite system $\mathcal{R}$ is a total quasi-order $\succeq$ on clean cyclic drags whose strict part is well-founded. A head order is said to be*

(i) compatible*: if two cyclic drags that are isomorphic up to their lists of roots are equivalent in the quasi-order;*
(ii) subcyclic*: if $W = A \otimes_\xi B$, and B is not isomorphic to an identity drag, then $W \succ A$;*
(iii) cyclic monotonic*: if for all $L \to R \in \mathcal{R}$, then for all minimal cyclic extensions (M, ξ) of L, $L \otimes_\xi M \succeq R \otimes_\xi M$ implies that for all cyclic extensions (E, ξ) of L, $L \otimes_\xi E \succ R \otimes_\xi E$.*

Compatibility includes cyclic-drag isomorphism strictly. Further, note that condition (iii) is global, quantifying separately on minimal cyclic extensions and nonminimal ones. A better definition of minimal cyclic extensions might allow us to weaken (iii).

Cyclic monotonicity tells us that the order between rules is preserved by growing the cycles generated by its minimal cyclic extensions. Note also that monotonicity and cyclic monotonicity are complementary: the former extends a drag arbitrarily, while preserving the original drag while the latter extends a drag by modifying its existing cycles only.

Definition 3.2 (GPO)**.** *Given two drags s, t and a head order $\succeq$, we define $s \succ t$ (under 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 3.3. *Consider the goal* $(\{1,2\},1,L(1) = L(2) = f, X(1) = 2$ *and* $X(2) = 1) \succ (\{1,2\},1,L(1) = f$ *and* $L(2) = a, X(1) = 2)$*, which corresponds to the rule of the (breaking) case in Figure 6. The first drag is a head. The head of the second is the drag* $(\{1,2\},1,L(1) = f$ *and* $L(2) = x', X(1) = x', \{2\})$*, which is a subdrag of the first (exercise for the reader). By subterm property of a head order, the first is strictly larger than the second in* $\geq$*. Therefore, we must now prove the subgoal:* $(\{1,2\},1,L(1) = L(2) = f, X(1) = 2$ *and* $X(2) = 1) \succ (\{1\},1,L(1) = a,)$*. This time, the second drag is a head itself, but is not a subdrag of the first. Therefore, we must have* $(\{1,2\},1,L(1) = L(2) = f, X(1) = 2$ *and* $X(2) = 1) \succcurlyeq (\{1\},1,L(1) = a,)$ *for this goal to succeed. As we shall see, such a head order is easy to define.*

4. 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 4.1 (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 (nonempty) head. □

Lemma 4.2 (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 4.3 (Subterm)**.** $\rhd \subseteq \succ$*.*

Proof. It is clear that a switchboard for (W,U) defines by restriction a switchboard for $(W,\nabla U)$, for which we keep the same name. Then, by Case (∇) of Definition 3.2, $U \succ \nabla U$. □

Lemma 4.4 (Variables)**.** *If* $U \succ V$*, then* $\mathcal{V}ar(U) \subseteq \mathcal{V}ar(V)$*.*

Proof. This is a well-known property of RPO-like definitions. □

Definition 4.5. *Let* U, V *and* $W = (V, R, L, X, S)$ *be open drags such that* $\mathcal{V}ar(V) \subseteq \mathcal{V}ar(U)$*, Two switchboards* ξ, ξ' *for* (W,U) *and* (W,V) *are* coherent *if* $\forall s,t,n,p.\ \xi_U(s) = n$*,* $\xi'_V(t) = p$*,* $L(s) = x$*, and* $L(t) = x$ *imply* $\mathcal{R}(W)(n) = \mathcal{R}(W)(p)$*.*

Lemma 4.6. *Let* U, V, W *be open drags such that* $U \succ V$*, and* ξ, ξ' *be two coherent switchboards for* (W,U) *and* (W,V) *respectively. Then,* ξ *is a switchboard for* (W,V) *such that* $W \otimes_\xi V = W \otimes_{\xi'} V$*.*

Proof. By Lemma 4.4, $\mathcal{V}ar(V) \subseteq \mathcal{V}ar(U)$. Then, coherence allows us to use ξ for ξ'. □

Theorem 4.1 (Transitivity)**.** *GPO ($\succ$) is transitive.*

Proof. Assuming that $u \succ t \succ s$, we show that $u \succ s$ by induction on (u, t, s) compared in $(\rhd)_{mul}$. Apart from the induction ordering itself, the proof is standard. □

We show compatibility of GPO with drag isomorphism. This result is of course important since it justifies our way of building compatibility into a path ordering via a careful definition of subdrags instead of using a normal-form representation of congruence classes of drags.

Theorem 4.2 (Compatibility)**.** *GPO is a strict ordering compatible with drag isomorphism.*

Proof. Antisymmetry follows from well-foundedeness proved next, and transitivity from Theorem 4.1. Compatibility follows from Theorem 2.1, since the definition of the order uses only heads and tails of the drags to be compared. □

Let *SN* stand for the strongly normalizing (terminating, well-founded) property of an element.

Lemma 4.7. *If the drag ∇t is SN, then the drag $t = g(\overline{t})$ is SN.*

Proof. The proof follows closely the proof schema initiated in [6]. By definition, t is SN iff all its reducts are SN. We prove that a given reduct v of t is SN by induction on the triple $\langle t, \nabla t, v\rangle$, compared in the lexicographic composition $\ggg \stackrel{\text{def}}{=} (\trianglerighteq, \succ, \rhd)_{lex}$. For this composition to be well-founded, each component relation must be well-founded. First, $\trianglerighteq$ is well-founded by definition of a head order. Second, $\rhd$ is well-founded by Lemma 4.3. Third, ∇t is SN by assumption. And since statuses preserve well-foundedness, $\succ$ is well-founded on lists, sets or multisets of elements taken from the set of elements smaller or equal to elements in ∇t. Therefore, $\ggg$ is well-founded. There are two kinds of reducts whose triple are smaller than $\langle t, \nabla t, v\rangle$: the reducts of t smaller than v in $\rhd$; the reducts of some v such that the pair $\langle v, \nabla v\rangle$ is smaller strictly than the pair $\langle t, \nabla t\rangle$ in the order $(\trianglerighteq, \succ)_{lex}$. 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:

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

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

Theorem 4.3 (Well-Foundedness)**.** *GPO is well-founded.*

Proof. Let $t = f(\overline{t})$. We show that t is SN by induction on $\rhd$. Since $t \rhd \nabla t$ by definition of $\rhd$, ∇t is SN by induction hypothesis. By Lemma 4.7, t is SN. The result follows. □

Theorem 4.4 (Totality)**.** *If the head order is total, 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 $\widehat{t} \succcurlyeq \widehat{v}$ or $\widehat{v} \succcurlyeq \widehat{t}$, then $t \succ v$ or $v \succ t$, respectively, by Case $\succcurlyeq$ of GPO. We are therefore left with the case where $\widehat{t} \doteq \widehat{v}$, for which we simply need to show that $\nabla t \succ_f \nabla v$ or $\nabla v \succ_v \nabla t$. 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 2.1, ensuring that one drag is bigger than the other. □

In the traditional term rewriting setting, a tree has a unique root; hence a rewrite rule is a pair of trees that have the name number of roots, namely one, on each side. This determines the notion of monotonicity of rewriting orders, which is then defined with respect to contexts with a single hole (or sprout). In the case of drags, a rewrite rule is a pair of drags with the same number of roots so as to be plugged in a context with (at least) that number of sprouts. Since the order is compatible with graph isomorphism, we can always assume that the drags are clean. Furthermore, the context must keep all roots originating from the drag to be rewritten, so that any vertex of the rewrite rule's left-hand side is accessible from one of them. This assumption is essential to obtain monotonicity, which, as we have seen, splits into two different notions, monotonicity and cyclic monotonicity, that we show in turn.

Theorem 4.5 (Monotonicity)**.** *GPO is monotonic.*

Proof. Let $\mathcal{R}$ be a drag rewrite system, $L \in R \in \mathcal{R}$ and ξ a directed rewriting switchboard for (W, L). Then, $W \otimes_\xi L \succ W \otimes_\xi R$. By Lemma 2.26, ξ is also a directed rewriting switchboard for (W, R).

The proof is by induction on $|W|$, where the size of a drag is the number of its internal vertices. By Theorem 4.2, we can assume without loss of generality that L, R are clean. If $|W| = 0$, W is an empty drag and the result follows from Lemmas 2.9 and 2.10, and Theorem 4.2. If $|W| \neq 0$, then

$W = \widehat{W} \otimes_\zeta \nabla W$, where $|\widehat{W}| \neq 0$ and ζ is a directed switchboard. It follows that $|W| > |\nabla W|$.

By Lemma 2.12, $W \otimes_\xi L = \widehat{W} \otimes_\zeta (\nabla W \otimes_\xi L)$ 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 2.1. By 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. □

This proof is of course similar to the monotonicity proof of RPO.

We are now left with cyclic monotonicity. Unfortunately, we have not been able yet to determine a finite set of minimal cyclic extensions of L that would suffice for GPO. Such extensions would satisfy the property that GPO behaves on an arbitrary extension as it behaves on a minimal one. It remains to be seen whether, and under what conditions, such minimal cyclic extensions exist.

5. Ordering Drag Heads

In this section, we define several different head orders. The first will be total but not monotonic. The second will be subcyclic and cyclic-monotonic, but not total. Both are defined by, first, interpreting a (cyclic) drag in terms of the function symbols labeling the internal nodes of the drag, and second generate an order on these interpretations from a total order on the set of function symbols. Both orders will be defined on arbitrary drags for convenience.

Let $\geq$ be a total order on Σ, called the *precedence*, whose strict part is well-founded, and $U = (V, R, L, X)$, a clean drag from which the sprouts and their incoming edges have been removed.

Total head order. We represent a drag as a list of function symbols.

Definition 5.1 (Interpretation)**.** *The* interpretation *of* $U = (V, R, L, X)$*, written* $[\![U]\!]$*, is a list of symbols in* Σ *defined as follows: If* $\mathcal{A}cc(U) = \varnothing$*, then* $[\![U]\!] \stackrel{\text{def}}{=} ()$*. Otherwise,* $[\![U]\!] \stackrel{\text{def}}{=} L(r) :: [\![W]\!]$*, where* $R = r :: R'$ *and* $W = (V \setminus r, (X(r) \setminus r) :: R', L, X')$*,* X' *being* X *with references to* r *eliminated.*

In words, $[\![U]\!]$ collects the function symbols labeling the internal nodes of a drag, by traversing this drag in a depth-first search manner starting from the list of roots of the drag.

We can now define our head order on drags:

Definition 5.2. $D \geq D'$ *iff* $\langle |[\![D]\!]|, [\![D]\!] \rangle (\geq_{\mathbb{N}}, \geq_{lex}) _{lex} \langle |[\![D']\!]|, [\![D']\!] \rangle$.

Lemma 5.3. $\geq$ *is a total head order.*

Proof. First, $\geq$ is an order whose strict part is clearly well-founded, since built from other well-founded orders by using lexicographic compositions. This is the reason why we use the length of the interpretation first. Furthermore, it is total since the precedence on function symbols is total. Finally,

it is routine to show that isomorphic drags have the same interpretation; hence the total order is compatible with drag isomorphism. □

Example 5.4. *Compare the drags of the first figure.*

Note that this head order is not subcyclic nor cyclic monotonic.
Subcyclic, cyclic-monotonic head order. We represent a drag as the multiset of function symbols labeling its accessible vertices:

Definition 5.5. *We define the* interpretation *of U, written $[\![U]\!]$, as the multiset of symbols in Σ that label the vertices of $U{\downarrow}$.*

Given two drags, we define:

Definition 5.6. *$D \trianglerighteq D'$ iff $[\![D]\!] \geq_{mul} [\![D']\!]$.*

Lemma 5.7. *$\trianglerighteq$ is a subcyclic, cyclic monotonic head order.*

Proof. By the same token as before, $\trianglerighteq$ is an order whose strict part is well-founded and compatible. Furthermore, the order is subcyclic, since adding new vertices in a drag increases the multiset of function symbols labeling its vertices. Finally, it is cyclic-monotonic, since increasing a minimal cyclic extension increases equally the interpretations of the two compared drag heads. □

Example 5.8. *Figure 7 highlights some comparisons generated when checking termination of the rule corresponding to the first comparison in the upper-left corner. Checking minimal directed extensions is easy; there is only one, which merges the two sprouts of the right-hand side of the rule. The precedence $f > g$ suffices for both comparisons. The other three extensions shown are cyclic. The comparison shown in the right upper corner succeeds by Case ∇ followed by Case $\trianglerighteq$. The other two comparisons succeed by Case $\trianglerighteq$.*

We obtain successively: $\{f, k\} >_{mul} \{g\}$ (upper left) and use of Case ∇ for the subterm; Case ∇, then $\{f\} >_{mul} \{g\}$ (upper right); $\{f, g, k\} >_{mul} \{g, g\}$ (lower left); and $\{f\} >_{mul} \{g\}$ (lower right). In all cases but the first, the right-hand side drag never has a subdrag.

These comparisons do not cover all possible cyclic extensions. Consider for example a cyclic extension obtained from the one in the upper right by mapping x to a bigger drag. Then, the comparisons obtained after the ∇ case may fail.

6. Conclusion

We have invented drags and explored their nice and useful properties. Furthermore, we have designed an ordering, GPO, that can be used for comparing drags, which 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,

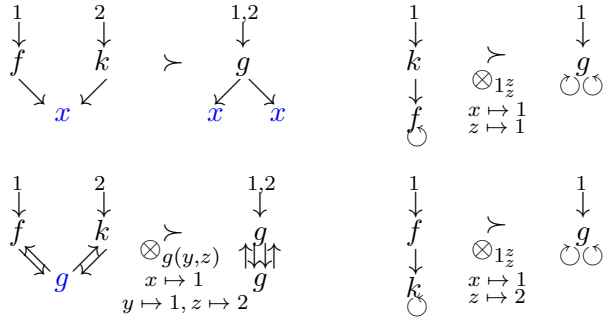

FIGURE 7. Checking termination.

e.g. operadic expressions. The second yields a monotonic order over drags in special cases.

Inventing additional monotonic variants, or finding useful special cases that ensure monotonicity, is now our first order of priority. We need to understand under exactly what conditions on the head order GPO becomes cyclic monotonic. Clearly, for dags – with sharing but sans cycles – all is well.

Our graph rewriting model does not include copying, hence does not truly generalize the term rewriting model. We conjecture that GPO scales to this richer framework.

The graph path ordering reuses Rubio's idea [8] of building a desired congruence over expressions: associativity and commutativity of some function symbols labeling trees in his case, and isomorphism of drags in our case. This is achieved via a definition of subterm that is invariant under the congruence. 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.

Our term representation and ordering have the potential to be further extended with abstraction and application, as done in the computability path order [3], and obtain an ordering over λ-terms with sharing and back-arrows, also called *lambda graphs* [1]. We presume that this problem conceals some unseen difficulties.

Our implementation generalizes previous work by allowing for many roots and will be fully described in a forthcoming paper.

REFERENCES

[1] Zena M. Ariola and Jan Willem Klop. Cyclic lambda graph rewriting. In *Proceedings of the Ninth Annual Symposium on Logic in Computer Science (LICS '94), Paris, France, July 4-7, 1994*, pages 416–425. IEEE Computer Society, 1994.

[2] László Babai. Graph isomorphism in quasipolynomial time [Extended abstract]. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, June 18-21, 2016*, pages 684–697, 2016.

[3] Frédéric Blanqui, Jean-Pierre Jouannaud, and Albert Rubio. The computability path ordering. *Logical Methods in Computer Science*, 11(4), 2015.
[4] Nachum Dershowitz. Orderings for term-rewriting systems. *Theor. Comput. Sci.*, 17:279–301, 1982.
[5] Vladimir Dotsenko and Murray Bremner. *Algebraic Operads: An Algorithmic Companion*. CRC Press, 2016.
[6] Jean-Pierre Jouannaud and Albert Rubio. Polymorphic higher-order recursive path orderings. *J. ACM*, 54(1), 2007.
[7] Yves Lafont. Algebra and geometry of rewriting. *Applied Categorical Structures*, 15(4):415–437, 2007.
[8] Albert Rubio. A fully syntactic AC-RPO. *Inf. Comput.*, 178(2):515–533, 2002.

School of Computer Science, Tel Aviv University

LIX, École Polytechnique and INRIA-Saclay