Drags: A Simple Algebraic Framework For Graph Rewriting

Nachum Dershowitz
School of Computer Science
Tel Aviv University, Tel Aviv, Israel
nachum.dershowitz@cs.tau.ac.il

Jean-Pierre Jouannaud
LIX & DEDUCTEAM
École Polytechnique, Palaiseau, France
jeanpierre.jouannaud@gmail.com

We are interested in a natural generalization of term-rewriting techniques to what we call *drags*, viz. finite, directed, ordered, rooted multigraphs. Each vertex is labeled by a function symbol, the arity of which governs the number of vertices to which it relates by an incoming edge in the graph. To this end, we develop a rich algebra of drags that generalizes the familiar term algebra and its associated rewriting capabilities. Viewing graphs as terms provides an initial building block for rewriting with such graphs, one that should impact the many areas where computations take place on graphs.

1 Introduction

Rewriting with graphs has a long history in computer science, graphs being used to represent data structures, but also program structures and even computational models. Since graphs originate from topological investigations, it is no wonder to see them also play an important rôle in modern algebraic topology, more specifically in the theory of operads [4]. Our interest in graphs originated from the latter, which has strong relationships with various areas of computer science, including concurrency theory and type theory.

As rewriting graphs is very similar to rewriting algebraic terms, the same questions recur: What rewriting relation do we use? Is there an efficient pattern matching algorithm? Is a particular rewriting system terminating? Is it confluent?

The above questions have therefore been addressed by the rewriting community since the mid-eighties. Termination and confluence techniques have been elaborated for various generalization of trees, such as rational trees, directed acyclic graphs, lambda-terms and lambda-graphs. But the design of a convenient mathematical framework for rewriting arbitrary graphs has made little progress beyond various categorical definitions and the study of the particular cases we just mentioned [9, 11]. See [8] for a survey of implementations of forms of graph rewriting and analysis tools.

This paper describes the design of a general class of graphs – actually, multigraphs – that has good structural properties with respect to graph rewriting. A specificity used for that purpose is the presence of arbitrarily many roots. In particular, graphs in this class can be equipped with a natural, simple algebraic structure that allows one to view them as terms with sharing and, therefore, to mimic many techniques that have been developed for trees in the past.

Despite being purely theoretical, the framework developed here should lead to easy implementations: the view of a drag as a term with sharing is extremely easy to implement, and therefore suggests the possibility to convert many existing term-rewriting implementations to actually do graph rewriting.

The particular class of graphs we are interested in is introduced in Section 2. The algebra of drags is described in Section 3, and the view of drags as terms in Section 4. Drag rewriting is introduced in Section 5.

Submitted to:
TERMGRAPH 2018

2 Drags

The class of graphs with which we deal here is that consisting of finite directed graphs with labeled vertices, allowing multiple edges between vertices and an arbitrary number of roots. 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, in pictures) 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. This class of graphs appears to be a very general model for describing computations in the absence of binding constructs.

To lessen notational burden, we 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 list concatenation as well as set and multiset union, $\setminus$ for set or multiset difference, and identify an empty or singleton list, set or multiset with its contents.

In contrast with the standard categorical approaches (see, e.g. [7, 10, 5, 6]), our multigraph model is very standard. The novelty, however, is that we allow for arbitrarily many roots and arbitrarily complex cycles: there is no restriction on the structure of the drags we deal with.

Definition 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*, such that $R(n)$ 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 $|L(v)|$. Labeling extend to lists, sets and multisets of vertices as expected.*

Given $b \in X(a)$, then (a,b) is an 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 *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'. We denote by R_{max} the set of maximal roots.*

Definition 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*. When nonempty, the set S of sprouts will be added at the end of the tuple which becomes $\langle V,R,L,X,S\rangle$. An open drag is* linear *if different sprouts have different labels. It is* cyclic *if $R \cap S = \varnothing$ and $\forall v \in V \setminus S.\ \exists r \in R_{max}.\ vX^*r$.*

The component R is a list with repetitions, whereas S is a set. Allowing for repeated roots is needed for rewriting. Sprouts may be roots. Cleanness relates to garbage collection, discussed further in Section 5. It will sometimes be convenient to consider roots as specific incoming edges and to identify a sprout with its label.

A cyclic drag does not necessarily have a cycle going through all its vertices, for at least three reasons: first, cyclicity is closed under union; second, its sprouts – if any – have no outgoing edges; third, an isolated root is cyclic by itself. A cyclic drag may therefore consist of a single internal vertex that is a root. Although this may seem odd to call such a drag cyclic, this is an essential ingredient here, which allows one to view a tree as a particular drag. For example, the ground term a and the term $f(x)$ are cyclic drags, while the terms $f(a)$ and $f(f(x))$ are 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.

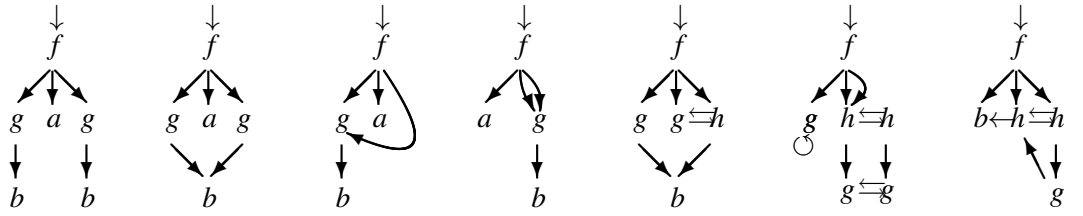

Figure 1: Seven non-isomorphic drags whose vertices are indicated by the labels.

Given a drag D, we use the following notations: $\mathscr{V}er(D)$ for its set of vertices; X_D for its successor function; $\mathscr{A}cc(D)$ for its set of accessible vertices; $\mathscr{R}(D)$ for its set of roots; $\mathscr{S}(D)$ for its set of sprouts; $\mathscr{V}ar(D)$ for the set of variables labeling its sprouts; $D^{\vec{r}}_{\vec{s}}$ to indicate its roots and sprouts; and $D^\sharp$ for the clean drag obtained by removing its inaccessible vertices and related edges. We denote by $L \setminus V$ the sublist of the list L of vertices obtained by filtering out those belonging to the set V.

Definition 3 (Subdrag)**.** *Given a drag $U = \langle V,R,L,X,S\rangle$ and a subset W of its vertices, 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 $(R\cap V')\cup(X(V\setminus V')\cap V')$.

A subdrag is clean by construction. Its roots are obtained by adding as new roots those vertices in V' that have an incoming edge in U that is not in $U_{|W}$, with the corresponding order of multiplicity. The order of elements in R' does not really matter for now.

In particular, restricting a drag D to its maximal roots, yields $D^\sharp$. On the other hand, restricting any drag to an empty set of vertices, yields the empty drag $\varnothing \stackrel{\text{def}}{=} 1^{\varnothing}_{\varnothing} = \langle \varnothing,\varnothing,\varnothing,\varnothing,\varnothing\rangle = \langle ,,,\rangle$.

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 4 (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* $\mathrm{o} : \mathscr{A}cc(V) \mapsto \mathscr{A}cc(V')$ *such that*

(i) o *restricts to bijections between the lists of roots and sets of accessible sprouts;*

(ii) $\forall v \in \mathscr{A}cc(V)\setminus S.\ L(v) = L'(\mathrm{o}(v))$ and $X(v) = X'(\mathrm{o}(v))$;

We write $D =_{\mathrm{o}} D'$, or simply $D = D'$ when D and D' are isomorphic drags. We write $D \simeq_{\mathrm{o}} D'$, 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 $D^\sharp$. 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 5. *Our running example is made up of seven single-rooted drags depicted 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 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* $|f|=3$, $|h|=2$, $|g|=1$, $|a|=|b|=0$. *We first consider the term* $f(g(b),a,g(b))$ *in which* b *and* $g(b)$ *can be shared. Among all possible cases, three 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 inner cycle.*

3 Drag Algebra

We equip drags with a composition operator, which, given two drags, combines them by connecting the sprouts of each to the roots of the other. This will be a central notion for us. Though the details may appear messy, the idea is quite intuitive: Take the union of the drags, connecting the sprouts and roots according to a device we call a *switchboard*.

As usual, we denote by $\mathscr{D}om(\xi)$ and $\mathscr{I}m(\xi)$ 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. Both are sets. Hence $\mathscr{I}m(\xi)$ may differ from $\xi(R)$, which is a list possibly with repetitions when R is a list itself.

We begin with the connection device:

Definition 6 (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 (we also say that* (D',ξ) *is an* extension *of* D*), such that*

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

(ii) $\forall s,t\in S'.\ s\in\mathscr{D}om(\xi_{D'})$ *and* $L'(s)=L'(t)$ *imply* $t\in\mathscr{D}om(\xi_{D'})$ *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 variadic vertices, isomorphism would be quasi-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 $\mathscr{V}ar(D)\cap\mathscr{V}ar(D')=\varnothing$ so as to avoid confusion, but this is by no means necessary since switchboards are pairs of mappings operating on $\mathscr{V}ar(D)$ and $\mathscr{V}ar(D')$ independently, not a single mapping operating on $\mathscr{V}ar(D)\cup\mathscr{V}ar(D')$. Further, splitting ξ makes the definition clearly symmetric.

Note finally 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. Further, injectivity implies that the list $\xi_D(\mathscr{D}om(\xi_D))$ is indeed a set, hence making the set difference $[1\,..\,|R'|]\setminus\xi_D(\mathscr{D}om(\xi_D))$ well defined.

Two particular kinds of switchboards play an important rôle:

Definition 7 (Directed Switchboard). *A switchboard ξ for (D,D') is* directed *if one of ξ_D and $\xi_{D'}$ has an empty domain; the pair $(D',\xi_{D'})$ is called a* context extension *of D if $\mathscr{D}om(\xi_D)=\varnothing$ and a* substitution extension *of D if $\mathscr{D}om(\xi_{D'})=\varnothing$.*

Definition 8 (Rewriting Switchboard). *A switchboard ξ for (D,D') is a* rewriting *switchboard if $\xi_{D'}$ is linear and surjective and ξ_D is total; the pair (D',ξ) is the* rewriting extension *of D.*

Directed switchboards correspond to the tree case, with all connections going from one drag to the other. Rewriting switchboards allow one to "encompass" drag D' within drag D. Note that context extensions and substitution extensions exchange each other by symmetry of the definition of a switchboard.

A switchboard induces a binary operation on open drags:

Definition 9 (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 $(V\setminus\mathscr{D}om(\xi_D))\cap(V'\setminus\mathscr{D}om(\xi_{D'}))=\varnothing$. Their* composition *is the drag $D\otimes_\xi D'\stackrel{\text{def}}{=}\langle V'',R'',L'',X'',S''\rangle$, where*

(1) $V''=(V\setminus\mathscr{D}om(\xi_D))\cup(V'\setminus\mathscr{D}om(\xi_{D'}))$;

(2) $R''=R([1\,..\,|R|]\setminus\xi_{D'}(S'))\setminus\mathscr{D}om(\xi_D)\;\cup\;R(\xi_{D'}(R'\setminus(V'\setminus S')))\;\cup\;R'([1\,..\,|R'|]\setminus\xi_D(S))\setminus\mathscr{D}om(\xi_{D'})\;\cup\;R'(\xi_D(R\setminus(V\setminus S)))$;

(3) $L''(v\in V\setminus\mathscr{D}om(\xi_D))=L(v)$ and $L''(v\in V'\setminus\mathscr{D}om(\xi_{D'}))=L'(v)$;

(4a) $X''(v\in V\setminus S)=$ if $X(v)\notin\mathscr{D}om(\xi_D)$ then $X(v)$; otherwise $R(\xi_D(v))$;

(4b) $X''(v'\in V'\setminus S')=$ if $X'(v')\notin\mathscr{D}om(\xi_{D'})$ then $X'(v')$; otherwise $R'(\xi_{D'}(v'))$;

(5) $S''=(S\setminus\mathscr{D}om(\xi_D))\cup(S'\setminus\mathscr{D}om(\xi_{D'}))$.

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''. This list is made of two sublists for each component drag; we describe them for D:

- The remaining roots of D: these are obtained by removing from R the roots that are in the image of $\xi_{D'}$ (note that we remove the root indices from the whole set of indices by making a set difference before to apply R), before filtering out all those sprouts belonging to the domain of ξ_D.
- The vertices of D belonging to the image by $\xi_{D'}$ of a sprout of D that was a root in D': vertices inherit the root status from the sprouts with which they are merged, with the appropriate order of multiplicity. (For that reason, $\xi_{D'}$ is applied here to a list of sprouts obtained from the list of roots by filtering out those which are not sprouts.)

  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, possibly in a cycle. In such cases, all need to be renamed to the same one, while accumulating root statuses.

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 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.

Example 10. *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*

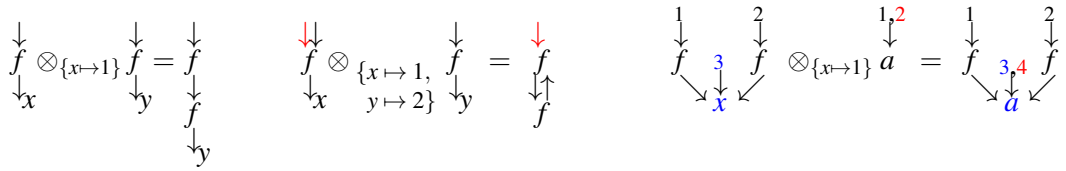

Figure 2: Unidirectional, cyclic and root-transfer compositions.

(red) root of the first drag which has two roots, the first red, the other black. The third shows how sprouts that are also roots connect to roots in the composition (colors indicate roots' origin). Note that the first root of the term a has disappeared in the composition, while its second root is now the 4th of the result.

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.

Lemma 11 (Commutativity)**.** *Composition of drags is quasi-commutative, and becomes commutative for switchboards whose context or substitution is surjective, such as 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 of the composition.

Let us call *identity* a linear open drag all of whose vertices are its sprouts and set of edges is empty. We denote it by 1_Z^Y, where $Y \subseteq Z^*$ is its list of roots. We use $\varnothing$ for the drag $1_Z^\varnothing$, which is isomorphic to $1_\varnothing^\varnothing$, and is called the empty drag for that reason.

Lemma 12 (Neutral)**.** *Let D be an open drag, $X \subseteq \mathscr{V}ar(D)$, 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 13 (Compatibility)**.** *Two switchboards ξ and ζ for the respective pairs of drags (U,V) and (V,W) are* compatible *if (i) $\mathscr{D}om(\xi_V) \cap \mathscr{D}om(\zeta_V) = \varnothing$ and (ii) $\mathscr{I}m(\xi_U) \cap \mathscr{I}m(\zeta_W) = \varnothing$.*

Lemma 14 (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. □

4 Drag Decomposition

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

Lemma 15. *Given a drag D and a subset of vertices $W \subseteq V$, there exists a drag A, called* antecedent*, and a directed switchboard ζ such that $D = A \otimes_\zeta D_{|W}$.*

Proof. A has for internal vertices those of D that do not belong to $D_{|W}$. Its sprouts are the sprouts of D that are not in $D_{|W}$, plus new sprouts that correspond to the new roots of $D_{|W}$, that is, to the vertices of $D_{|W}$ whose antecedents in D are not vertices of $D_{|W}$. These sprouts can be labeled by different variables

so as to make A linear on them. The roots of A are those of D that are not vertices of $D_{|W}$, plus the new sprouts that become roots with the appropriate order of multiplicity (the corresponding number of outgoing edges to $D_{|W}$ in D). The switchboard ξ mapping these sprouts to the corresponding roots of $D_{|W}$ is directed since subdrags are closed under successor and total on them. Note that the rooted-sprouts of A disappear in the composition. □

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 by definition of a subdrag. Note also that ξ induces an order on the roots of the subdrag that are not roots of the whole drag. If $W = V$, then (A, ξ) must be the identity extension 1_X^X with $|X| = |\mathscr{R}(D)|$.

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 that 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 – also the smallest nontrivial antecedent 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 that 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 16. *The* tail ∇D *of a drag* $D = \langle V, R, L, X, S \rangle$ *is the subdrag generated by the set of vertices* $V \setminus \{v \in V : vX^*\mathscr{R}(D)_{max}\}$.

As an application of Lemma 15, we get:

Lemma 17. *Given a drag D, there exists a 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. By Lemma 15, it contains all maximal roots of D. Note further that its sprouts that were not already sprouts of D are linear and that ξ is total on those sprouts. More precisely, ξ is a bijection between those sprouts 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 that 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. 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 18 (Structure)**.** *Isomorphic drags have isomorphic heads and tails.*

Proof. Given two isomorphic drags D, D', constructing bijections between their respective heads and tails from the bijection between their sets of vertices is an easy exercise. □

Example 19. *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.*

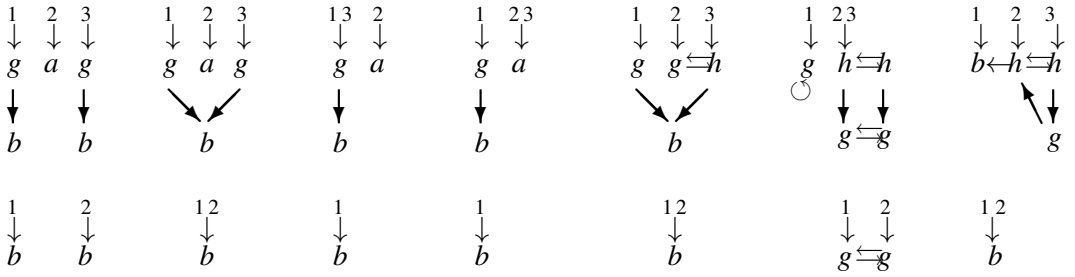

Figure 3: Successive subdrags of the drags of Figure 1

The fundamental property of a drag expressed by Theorem 18 is that its decomposition into a head and tail is a faithful representation of it. This property holds true because drags are multirooted. 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.

5 Drag Rewriting

The notion of composition lead in the previous section to a 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: 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 that is therefore a substitution extension of U; those that are not accessible from the sprouts of U form the remaining part A, which is therefore a context extension of U.

Particular substitution extensions of U are identity 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, and, of course, $z \notin \mathscr{D}om(\xi)$. Then, the drag resulting from the composition of U with that extension 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 z. The rôle of these identity extensions is to modify the structure of U by introducing sharing among its sprouts.

We are left defining the kind of extension that B belongs to:

Definition 20 (Cyclic extension)**.** *An extension (B, ξ) of a clean drag U is* cyclic *if B is generated by $\mathscr{I}m(\xi_U)$, $B \otimes_\xi U$ is a clean nonempty drag, and for all $t \in \mathscr{V}er(B)$, there exists $s \in \mathscr{D}om(\xi_B)$ such that $tX^*_D s$. The extension is* trivial *if B is an identity drag and* total *if ξ_B is surjective.*

This condition resembles the one for cyclic drags, but does not always imply that $B \otimes_\xi U$ is cyclic because not all internal vertices of U may reach a root of $B \otimes_\xi U$. Only *total* extensions are cyclic, up to possibly ground subdrags of U when there are such subdrags in U.

The conditions for being a cyclic extension impose that ξ_U is surjective on $\mathscr{R}(B)_{max}$ so as to generate B. The roots of the resulting drag are therefore originating from either B or U (possibly via a transfer), but there must be sufficiently many of them so that the resulting drag is clean.

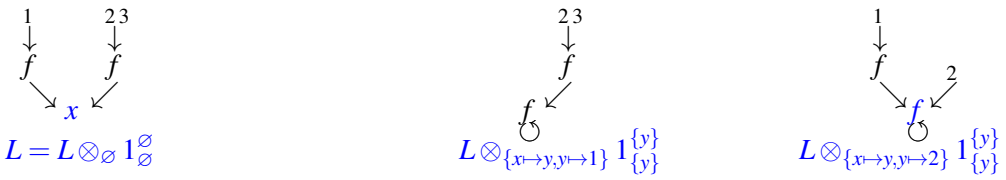

Figure 4: Identity cyclic extensions.

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 $\mathscr{D}om(\iota_U) \subseteq \mathscr{V}ar(U)$, 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. Unfortunately, identity cyclic extensions do not suffice for that purpose unless in special cases:

Example 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.*

Identity extensions 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 multirooted drags, since identity cyclic extensions cannot reach two different roots from the same sprout. Note that nonidentity cyclic extensions cannot do any better if the vocabulary contains unary symbols only, as in Example 21. If the vocabulary contains a symbol g with $|g| = 2$, then new cyclic extensions pop up, both with the same drag $(\{1,2,3\}, 1, \{1 \mapsto_L g, 2 \mapsto_L y, 3 \mapsto_L z\}, 1 \mapsto_X \{2,3\}, \{2,3\})$, and with the respective switchboards $\{x \mapsto 1, y \mapsto 1, z \mapsto 2\}$ and $\{x \mapsto 1, y \mapsto 1, z \mapsto 3\}$.

Here is the property at the heart of drag rewriting:

Lemma 22 (Decomposition)**.** *Let U, W be clean nonempty drags and ξ be a switchboard for (W, U). If ξ_W is surjective on $\mathscr{R}(U)_{max}$, then, there exist drags A, B, C and switchboards ζ, θ such that*

1. *$(B, \langle \xi_B, \xi_{U \to B} \rangle)$ is a cyclic extension of U denoted by (B, ξ);*
2. *(C, θ) is a substitution extension of $B \otimes_\xi U$;*
3. *(A, ζ) is a context extension of $(B \otimes_\xi U) \otimes_\theta C$;*
4. *$W \otimes_\xi U = A \otimes_\zeta ((B \otimes_\xi U) \otimes_\theta C)$;*
5. *C is empty if all internal nodes of W reach one of its sprouts;*
6. *$B \otimes_\xi U$ is cyclic if C is empty and ξ_B is surjective on $\mathscr{R}(U)_{max}$.*

Proof. The lemma is depicted in Figure 5. Let first D be the subdrag of $W \otimes_\xi U$ generated by $\mathscr{R}(U)_{max}$. Since U is clean, D must contain all vertices of U by definition of $\mathscr{R}(U)_{max}$. By Lemma 15, $W \otimes_\xi U = A \otimes_\zeta D$ for some context extension (A, ζ) of D such that ζ is total, surjective and directed. Observe that $\zeta_{A \to U} = \xi_A$. Let now C be the subdrag of D generated by those vertices that cannot reach a sprout in $\mathscr{D}om(\xi_W)$, hence a root of U. By definition of C and the fact that D contains all vertices of U, the ancestor of C in D must contain all vertices of U, hence is of the form $B \otimes_{\langle \xi_B, \xi_{U \to B} \rangle} U$. By Lemma 15, $D = (B \otimes_\xi U) \otimes_\theta C$ for some θ that is total, surjective and directed. Observe that $\theta_U = \xi_{U \to C}$. Property (4) now follows. Properties (5) and (6) are straightforward. □

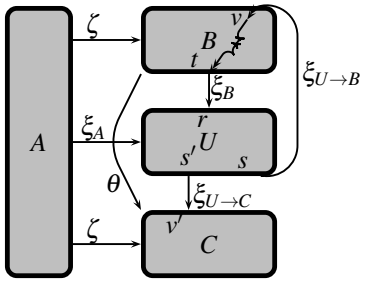

Figure 5: Decomposition of a pair of drags.

This lemma applies of course to rewriting extensions (W,ξ) of a drag U, since, in this case, ξ_W must be surjective on all roots of U.

Definition 23 (Rules)**.** *A* graph rewrite rule *is a pair of open clean drags written $L \to R$ such that $|\mathscr{R}(L)| = |\mathscr{R}(R)|$ and $\mathscr{V}ar(R) \subseteq \mathscr{V}ar(L)$. 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 $\mathscr{V}ar(R) = \mathscr{V}ar(L)$. Adding unreachable sprouts to a clean R would then do with an isomorphic right-hand side. Such a handy trick is not possible with trees.

Definition 24 (Rewriting)**.** *Let $\mathscr{R}$ be a graph rewrite system. We say that a nonempty clean drag D* rewrites *to a clean drag D', and write $D \longrightarrow_{\mathscr{R}} D'$ iff $D = W \otimes_\xi L$ and $D' = (W \otimes_\xi R)^\sharp$ for some drag rewrite rule $L \to R \in \mathscr{R}$ and rewriting extension (W,ξ) of L, such that ξ_L is linear if L is linear.*

All assumptions on ξ play a rôle. First, because ξ is a rewriting switchboard, ξ_W must be linear. This implies that the variables labeling the sprouts of W that 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.

In the case of trees, the rewrite extension of L is made of a context and a substitution. What is remarkable about the above definition is that there is no longer any distinction between the two, which, together, constitute the rewriting extension of the left-hand side. The context and substitution of L correspond respectively to the context extension and substitution extension of Lemma 22. The cyclic extension is of course the identity in the case of trees.

Example 25. *Consider the cyclic composition of Figure 2, and let $f(y') \to y'$ be the graph rewrite rule whose left-hand and right-hand sides are the drags $\langle\{1,2\},1,\{1 \mapsto_L f, 2 \mapsto_L y'\},1 \mapsto_X 2,2\rangle$ and $\langle 1,1,1 \mapsto_L y',,1\rangle$, respectively. The drag $D = \langle\{1,2\},1,\{1 \mapsto_L f, 2 \mapsto_L f\},\{1 \mapsto_X 2, 2 \mapsto_X 2\}\rangle$ rewrites*

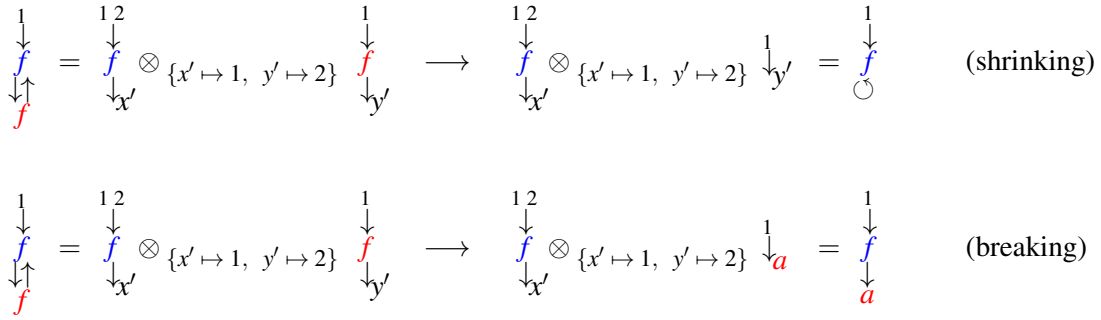

Figure 6: Rewriting a cycle.

to the drag $D' = \langle\{1,2\},1,\{1 \mapsto_L f, 2 \mapsto_L x'\},1 \mapsto_X 2,2\rangle \otimes_{\{x' \mapsto 1, y' \mapsto 1\}} \langle 1,1,1 \mapsto_L y',,1\rangle = \langle 1,1,1 \mapsto_L f,1 \mapsto_X 1\rangle$. *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 D *rewrites to the drag* $D'' = \langle\{1,2\},1,\{1 \mapsto_L f, 2 \mapsto_L x'\},1 \mapsto_X x',2\rangle \otimes_{x' \mapsto 1} \langle 1,1,1 \mapsto_L a\rangle = \langle\{1,2\},1,\{1 \mapsto_L f, 2 \mapsto_L a\},1 \mapsto_X 2\rangle$ *with the rule* $f(y') \mapsto a$, *the sides of which are the drags* $\langle\{1,2\},1,\{1 \mapsto_L f, 2 \mapsto_L y'\},1 \mapsto_X 2,2)$ *and* $(1,1,1 \mapsto_L a,\rangle$.

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

Lemma 26. *If* $U \longrightarrow_{\mathscr{R}} V$, *then* $\mathscr{V}ar(V) \subseteq \mathscr{V}ar(U)$.

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

Lemma 27. *Assume* U,V,W *are three open drags such that* U *rewrites to* V, *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 $\mathscr{V}ar(V) \subseteq \mathscr{V}ar(U)$, ξ_V is the restriction of ξ_U to the sprouts of V whose variables are in $\mathscr{V}ar(W)$. □

A consequence of this property is that properties of extensions are preserved by rewriting. Further, the decomposition lemma (Lemma 22) itself is also preserved:

Lemma 28. *Assume* $U \underset{\mathscr{R}}{\longrightarrow} V$. *Then,* $U = A \otimes_\zeta ((B \otimes_\xi L) \otimes_\theta C)$ *and* $V \equiv A \otimes_\zeta ((B \otimes_\xi L) \otimes_\theta C)$ *for some* A,B,C *and* ζ,ξ,θ *such that* (A,ζ), (B,ξ) *and* (C,θ) *are context, cyclic and substitution extensions, respectively.*

6 Conclusion

We have invented drags, a very general class of multigraphs, and explored their nice and useful properties, which yield a new, simple and very effective model for graph rewriting.

In a companion paper [3], we use this model to show that termination techniques that have been developed for terms, specifically, the recursive path order, scale to drags. This work shows that the difficulty lies in monotonicity with respect to cyclic extensions: breaking or forming cycles is indeed a new phenomenon that shows up with our drag model, but does not in the absence of cycles. Moreover,

we have succeeded in designing a total order for extending the theory of Gröbner bases to algebraic operads, which are polynomial expressions built over drags (making cyclic monotonicity vanish in this case).

One can ask whether our drag model can capture term rewriting per se? There is at least one simple way in which that could be achieved. Taking terms with sharing as our term structure, terms without sharing can be seen as normal forms in that richer structure. We could, therefore, encompass term rewriting within our model by changing the definition of a clean drag. Details remain to be worked out.

A pleasant extension of our framework would be to allow for symbols of varyadic arity in order to facilitate rewriting modulo associativity and commutativity. We have not tried this yet.

Our framework has the potential to be further extended with abstraction and application, in order to obtain a language for λ-terms with sharing and back-arrows, also called *lambda graphs* [1]. This problem may conceal some unseen difficulties.

References

[1] Zena M. Ariola & Jan Willem Klop (1994): *Cyclic Lambda Graph Rewriting*. In: *Proceedings of the Ninth Annual Symposium on Logic in Computer Science (LICS '94), Paris, France, July 4-7, 1994*, IEEE Computer Society, pp. 416–425, doi:10.1109/LICS.1994.316066. Available at http://dx.doi.org/10.1109/LICS.1994.316066.

[2] László Babai (2016): *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*, pp. 684–697, doi:10.1145/2897518.2897542. Available at http://doi.acm.org/10.1145/2897518.2897542.

[3] Nachum Dershowitz & Jean-Pierre Jouannaud (2018): *GPO: A Path Ordering for Graphs*. In: *Proceedings of the Sixteenth International Workshop on Termination (WST 2018)*, Oxford, UK.

[4] Vladimir Dotsenko & Murray Bremner (2016): *Algebraic Operads: An Algorithmic Companion*. CRC Press.

[5] Frank Drewes, Hans-Jörg Kreowski & Annegret Habel (1997): *Hyperedge Replacement, Graph Grammars*. In Rozenberg [11], pp. 95–162.

[6] H. Ehrig, G. Engels, H.-J. Kreowski & G. Rozenberg, editors (1999): *Handbook of Graph Grammars and Computing by Graph Transformation: Vol. 2: Applications, Languages, and Tools*. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

[7] Hartmut Ehrig, Michael Pfender & Hans Jürgen Schneider (1973): *Graph-Grammars: An Algebraic Approach*. In: *14th Annual Symposium on Switching and Automata Theory, Iowa City, Iowa, October 15-17, 1973*, IEEE Computer Society, pp. 167–180, doi:10.1109/SWAT.1973.11. Available at http://dx.doi.org/10.1109/SWAT.1973.11.

[8] Barbara König, Dennis Nolte, Julia Padberg & Arend Rensink (2018): *A Tutorial on Graph Transformation*. In: *Graph Transformation, Specifications, and Nets – In Memory of Hartmut Ehrig*, *Lecture Notes in Computer Science* 10800, Springer, pp. 83–104, doi:10.1007/978-3-319-75396-6_5. Available at https://doi.org/10.1007/978-3-319-75396-6_5.

[9] Detlef Plump (1997): *Simplification Orders for Term Graph Rewriting*. In Igor Prívara & Peter Ruzicka, editors: *Mathematical Foundations of Computer Science 1997, 22nd International Symposium, MFCS'97, Bratislava, Slovakia, August 25-29, 1997, Proceedings*, *Lecture Notes in Computer Science* 1295, Springer, pp. 458–467, doi:10.1007/BFb0029989. Available at https://doi.org/10.1007/BFb0029989.

[10] Jean-Claude Raoult (1984): *On Graph Rewritings*. *Theor. Comput. Sci.* 32, pp. 1–24, doi:10.1016/0304-3975(84)90021-5. Available at http://dx.doi.org/10.1016/0304-3975(84)90021-5.

[11] Grzegorz Rozenberg, editor (1997): *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. World Scientific.