Confluence of drag rewriting

Jean-Pierre Jouannaud[1] and Fernando Orejas[2]

1 **École Normale de Paris-Saclay, Laboratoire de Spécification et Vérification, France**
2 **Universitat Politècnica de Catalunya, Barcelona, Spain**

Abstract

We develop a confluence result for graph rewriting based on the drag model [5].

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 concurrent and distributed computational models. They therefore play a key rôle in program evaluation, transformation, and optimization, and more generally program analysis; see, for example, [2].

Our work is based on a recent, purely combinatorial view of graphs [5]. To assess our claim that drags are a natural generalization of terms, we extend the most useful term rewriting techniques to drags: the recursive path ordering [4]; unification [8]; confluence in this paper. Our main result is that confluence of a terminating set of drag rewrite rules can be decided by the usual joinability test of their critical pairs. Comparisons with the literature will be done at length in a forthcoming paper.

2 The drag model [5]

Drags are finite, ***d***irected, ordered, ***r***ooted, l***a***beled multi-***g***raphs. Vertices with no outgoing edges are designated *sprouts*. Other vertices are *internal*. We presuppose: a set of function symbols Σ, whose elements f, equipped with a fixed arity $|f|$, are used as labels for internal vertices; and a set of nullary variable symbols Ξ, disjoint from Σ, used to label sprouts.

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

1. V is a finite set of *vertices*, denoted by $\mathcal{V}er(D)$;
2. $R : [p \,..\, p + |R|] \to V$ is a finite list of vertices, called *roots*; $R(p + n - 1)$ refers to the nth root in R; unless otherwise stated, $p = 1$; we use $\mathcal{R}(D)$ for the roots of D;
3. $S \subseteq V$ is a set of *sprouts*, leaving $V \setminus S$ to be the *internal* vertices;
4. $L : V \to \Sigma \cup \Xi$ is the *labeling* function, mapping internal vertices $V \setminus S$ to labels from the vocabulary Σ and sprouts S to labels from the vocabulary Ξ, writing $v : f$ for $f = L(v)$;
5. $X : V \to V^*$ is the *successor* function, mapping each vertex $v \in V$ to a list of vertices in V whose length equals the arity of its label (that is, $|X(v)| = |L(v)|$).

The reflexive-transitive closure X^* of the relation X is called *accessibility*. Vertex v is *accessible* if it is accessible from some root. A drag is *clean* if all its vertices are accessible. Terms as ordered trees, sequences of terms (forests), terms with shared subterms (dags) and sequences of dags (jungles) are all particular kinds of clean rooted drags.

Sprouts may be roots, this is essential for having a nice algebra of drags.

We use $\mathcal{V}ar(D)$ for the set of variables labeling the sprouts of D. A drag is *linear* if no two sprouts have the same label, in which case variables and sprouts can be identified.

2.1 Drag composition

A variable in a drag should be understood as a potential connection to a root of another drag, as specified by a connection device called a *switchboard*. A switchboard ξ is a pair of partial injective functions, one for each drag, whose *domain* $\mathcal{D}om(\xi)$ and *image* $\mathcal{I}m(\xi)$ are a set of sprouts of one drag and a set of positions in the list of roots of the other, respectively.

▸ **Definition 2** (Switchboard)**.** Let $D = \langle V, R, L, X, S\rangle$ and $D' = \langle V', R', L', X', S'\rangle$ be 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 such that

1. $s \in \mathcal{D}om(\xi_D)$ and $L(s) = L(t)$ imply $t \in \mathcal{D}om(\xi_D)$ and $\xi_D(s) = \xi_D(t)$ for all sprouts $s, t \in S$;
2. $s \in \mathcal{D}om(\xi_{D'})$ and $L'(s) = L'(t)$ imply $t \in \mathcal{D}om(\xi_{D'})$ and $\xi_{D'}(s) = \xi_{D'}(t)$ for all $s, t \in S'$;
3. ξ is *well-behaved*: it does not induce any cycle among sprouts, using ξ, R, R' relationally:
$$\nexists\ n > 0, s_1, \ldots, s_{n+1} \in S, t_1, \ldots, t_n \in S', s_1 = s_{n+1}.\ \forall i \in [1..n].\ s_i\ \xi_D R' X'^*\ t_i\ \xi_{D'} R X^*\ s_{i+1}$$

The pair $\langle D', \xi\rangle$ is an *extension* of D, a *rewriting extension* if ξ_D is surjective and $\xi_{D'}$ total.

Sprouts labelled by the same variable should be connected to the same vertex, as stipulated by conditions (1,2). It follows that $\xi_D(\mathcal{D}om(\xi_D))$ must be a set, making the set difference $[1\,..\,|R'|] \setminus \xi_D(\mathcal{D}om(\xi_D))$ well defined.

We now move to the composition operation on drags induced by a switchboard. The essence of this operation is that the (disjoint) union of the two drags is formed, but with sprouts in the domain of the switchboards merged with the roots to which the switchboard images refer. Merging sprouts with their images requires one to worry about the case where multiple sprouts are merged successively, when the switchboards map sprout to rooted-sprout to rooted-sprout, until, eventually, an internal vertex of one of the two drags must be reached because a switchboard is well-behaved. That vertex is called *target*:

▸ **Definition 3** (Target)**.** Let $D = \langle V, R, L, X, S\rangle$ and $D' = \langle V', R', L', X', S'\rangle$ be drags such that $V \cap V' = \varnothing$, and ξ be a switchboard for D, D'. The *target* $\xi^*(s)$ is a mapping from sprouts in $S \cup S'$ to vertices in $V \cup V'$ defined as follows:

Let $v = R'(n)$ if $s \in S$, and $v = R(n)$ if $s \in S'$, where $n = \xi(s)$.

1. If $v \in (V \cup V') \setminus (S \cup S')$, then $\xi^*(s) = v$.
2. If $v \in (S \cup S') \setminus \mathcal{D}om(\xi)$, then $\xi^*(s) = v$.
3. If $v \in \mathcal{D}om(\xi)$, then $\xi^*(s) = \xi^*(v)$.

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

We are now ready for defining the composition of two drags. Its set of vertices will be the union of two components: the internal vertices of both drags, and their sprouts which are not in the domain of the switchboard. The labeling is inherited from that of the components.

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

1. $V'' = (V \cup V') \setminus \mathcal{D}om(\xi)$;
2. $S'' = (S \cup S') \setminus \mathcal{D}om(\xi)$;
3. $R'' = \xi^*(R([1\,..\,|R|] \setminus \xi_{D'}(\mathcal{D}om(\xi_{D'})))) \cup \xi^*(R'([1\,..\,|R'|] \setminus \xi_D(\mathcal{D}om(\xi_D))))$;
4. $L''(v) = L(v)$ if $v \in V \cap V''$; and $L''(v) = L'(v)$ if $v \in V' \cap V''$;
5. $X''(v) = \xi^*(X(v))$ if $v \in V \setminus S$; and $X''(v) = \xi^*(X'(v))$ if $v \in V' \setminus S'$

If $\langle D, \xi\rangle$ is a rewriting extension of D', then *all* roots and sprouts of D' disappear in the composed drag. The drag D can then be seen as the context of the left-hand side of a rule $D' \to R$, where R must have the same number of roots as D' (and $\mathcal{V}ar(R) \subseteq \mathcal{V}ar(D')$.)

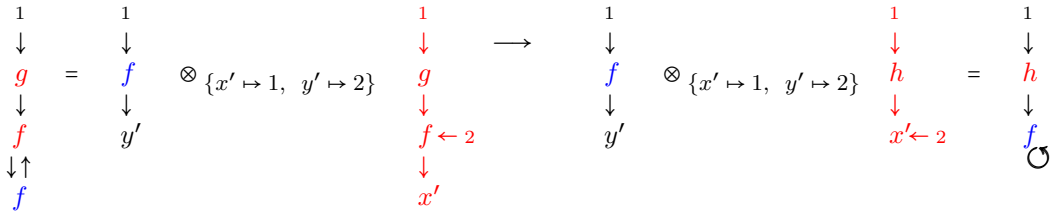

Figure 1 Rewriting and cycles.

2.2 Drag rewriting

Rewriting with drags is similar to rewriting with trees: we first select an instance of the left-hand side L of a rule in a drag D by exhibiting an extension $\langle W, \xi \rangle$ such that $D = W \otimes_\xi L$ – this is drag matching, then replace L by the corresponding right-hand side R in the composition. A very important condition for the result to be a drag is, accordingly, that the left- and right-hand sides of rules have the same number of roots:

▸ **Definition 5** (Rules). A *drag rewrite rule* is a pair of clean drags, written $\text{L} \to \text{R}$, such that (i) $|\mathcal{R}(\text{L})| = |\mathcal{R}(\text{R})|$, and (ii) $\mathcal{V}ar(\text{R}) \subseteq \mathcal{V}ar(\text{L})$. A drag rewriting system is a set of drag rewrite rules.

Condition (i) ensures that L and R have a perfect fit with any same environment, that is, both can be composed with any extension of L. Condition (ii) is standard for rewrite rules.

▸ **Definition 6** (Rewriting). Let $\mathcal{S}$ be a drag rewrite system. We say that a nonempty clean drag D *rewrites* to a clean drag D', and write $D \longrightarrow_{\mathcal{S}} D'$, iff $D = C \otimes_\xi \text{L}$ and $D' = C \otimes_\xi \text{R}$ for some drag rewrite rule $\text{L} \to \text{R} \in \mathcal{S}$ and clean rewriting extension $\langle C, \xi \rangle$ of L.

Because ξ is a rewriting switchboard, ξ_C must be linear, implying that the variables labeling the sprouts of C that are not already sprouts of D must all be different. Then, ξ_C must be surjective, implying that the roots of L (hence those of R) disappear in the composition, a case where the composition is commutative –we shall mostly write the context on the left, though. Further, ξ_{L} must be total, implying that the sprouts of L (hence those of R) disappear in the composition. Finally, D and C being clean, it is easy to show that D' is clean as well, which is therefore a property rather than a requirement.

▸ **Example 7** (Figure 1). The (red) rewrite rule $g(f(x')) \to h(x')$, whose roots are g and f on the left-hand side and h and x' on the right-hand side, applies with a blue context, colours which are reflected in the input and output terms (the rule applies across the cycle).

We are now finished with the material from [5] needed for the rest of this paper.

2.3 Drag unification [8]

The purpose here is to *identify* two clean drags U, V by composing them with the same *minimal* rewriting context $\langle C, \xi \rangle$, resulting in the same drag W. An identification corresponds to the fact that we want the same drag to be rewritten by two different rewrite rules whose left-hand sides are U and V. In order for $C \otimes_\xi U$ and $C \otimes_\xi V$ to both make sense, we assume that U, V are renamed apart (variables and root numbers).

▸ **Definition 8.** Given drags U, V, we call *partner vertices* two lists L_U, L_V of equal length of internal vertices of U and V, respectively, such that no two vertices $u, u' \in L_U$ (resp., $v, v' \in L_V$) are in relationship with X_U (resp., X_V).

▸ **Definition 9.** Two drags U, V are *identified* with a drag W at *partner vertices* $(\overline{u}, \overline{v})$ by an injective function $o : \mathcal{V}er(U) \cup \mathcal{V}er(V) \to \mathcal{V}er(W)$ called *identification*, written $U[\overline{u}] =_o V[\overline{v}]$, iff:
1. $o(\overline{u}) = o(\overline{v})$;
2. $\forall w \in \mathcal{V}er(U), w' \in \mathcal{V}er(V)$ such that $o(w) = o(w')$, $w : f$ iff $w' : f$ iff $o(w) = o(w') : f$;
3. $\forall w \in \mathcal{V}er(U), w' \in \mathcal{V}er(V)$ such that $o(w) = o(w')$, $o(X_U(w)) = o(X_V(w))$.

While two terms u, v are unified *at their root*, the solution being a substitution σ such that $u\sigma = v\sigma$, two drags U, V are unified at partner vertices $(\overline{u}, \overline{v})$, the solution being an extension $\langle C, \xi \rangle$ of both U and V that identifies $C \otimes_\xi U$ and $C \otimes_\xi V$ at these partner vertices:

▸ **Definition 10.** A *unification problem* is a pair of clean drags (U, V) that are renamed apart, together with partner vertices $P = \{(\overline{u}, \overline{v})\}$, which we write $U[\overline{u}] = V[\overline{v}]$. A *solution* (or *unifier*) to the unification problem $U[\overline{u}] = V[\overline{v}]$ is a clean rewriting extension $\langle C, \xi \rangle$ such that the *overlap* drags $C \otimes_\xi U$ and $C \otimes_\xi V$ are identified at P. A unification problem $U[\overline{u}] = V[\overline{v}]$ is *solvable* if it has a solution.

We want unification to be minimal, that is, to capture all possible extensions that identify U and V, without useless identifications occuring above or below partner vertices.

▸ **Definition 11.** We say that a drag U is an *instance* of a drag V, or that V *subsumes* U, and write $U \geq V$, if there exists a clean context extension $\langle C, \xi \rangle$ such that $U = C \otimes_\xi V$.

Cleanness is essential here, since otherwise any drag would be an instance of any other drag, which is also the reason why rewriting considers clean extensions only. In the following, we assume for convenience that the sprouts of U, V are labelled by different sets of variables.

▸ **Lemma 12.** *$\geq$ is a quasi-order whose equivalence is variable renaming and strict part is a well-founded order.*

We now extend the (of course well-founded) subsumption order to context extensions:

▸ **Definition 13.** Let U be a drag, of which $\langle C, \xi \rangle$ and $\langle D, \zeta \rangle$ are two context extensions. We say that (D, ζ) is an *instance* of (C, ξ) (or that (C, ξ) *subsumes* (D, ζ)) w.r.t. U, and write $(D, \zeta) \geq_U (C, \xi)$, if $(D \otimes_\zeta U)$ is an instance of $(C \otimes_\xi U)$.

We show in [8] the following key result:

▸ **Theorem 14.** *Given a unification problem, there is a unique most general unifying extension, if any, computable in quadratic time.*

3 Confluence

Confluence of a terminating term rewriting system follows from the joinability of its critical pairs, obtained by unifying overlapping left-hand sides of rules [9]. Our goal is to generalize this result to the drag framework. For non-terminating countable systems, replacing joinability by the existence of decreasing diagrams [10] should also work.

▸ **Lemma 15.** *Let $S \longleftarrow_{\mathrm{L}\to\mathrm{R}} U \longrightarrow_{\mathrm{G}\to\mathrm{D}} T$, and assume that U has no internal vertex being at the same time an internal vertex of* L *and of* G*. Then, there exist two drags V, W and a switchboard ξ such that $U = V \otimes_\xi W$, $V \longrightarrow_{l\to r} V'$, $W \longrightarrow_{\mathrm{G}\to\mathrm{D}} W'$, $S = V' \otimes_\xi W$ and $T = V \otimes_\xi W'$.*

Proof. By definition of rewriting, there exist rewriting extensions $\langle A, \xi \rangle$ and $\langle B, \zeta \rangle$ such that $U = A \otimes_\xi \mathrm{L} = B \otimes_\zeta \mathrm{G}$. We assume without loss of generality that L and G are renamed apart as well as A and B, making $\xi_\mathrm{L} \cup \zeta_\mathrm{G}$ well defined, as well $\mathcal{R}(A) \cup \mathcal{R}(B)$.

Let now C be the drag whose internal vertices are those of U which are not internal vertices of either L or G, its roots and sprouts are those of A plus those of B, and its successor relationship is inherited from that of U.

Since L and G do not share internal vertices by assumption, $B = \mathrm{L} \otimes_\xi C$ while $A = \mathrm{G} \otimes_\zeta C$, hence $U = \mathrm{L} \otimes_\xi C \otimes_\zeta \mathrm{G}$ (using associativity of composition [5]). We then take $V = \mathrm{L}$ and $W = C \otimes_\zeta \mathrm{G}$. ◂

▸ **Lemma 16** (Commutation). *Assume that* $U = V \otimes_\xi W$, $V \longrightarrow_{l\to r} V'$ *and* $W \longrightarrow_{g\to d} W'$. *Then,* $U \longrightarrow_{l\to r} V' \otimes_\xi W \longrightarrow_{g\to d} V' \otimes_\xi W'$ *and* $U \longrightarrow_{g\to d} V \otimes_\xi W' \longrightarrow_{l\to r} V' \otimes_\xi W'$.

Proof. Easy consequence of associativity of composition. ◂

▸ **Lemma 17** (Critical overlap). *Let* $S \longleftarrow_{L\to R} U \longrightarrow_{\mathrm{G}\to D} T$, *and let us assume that* U *has an internal vertex* w *which is an internal vertex of both* L *and* G. *Then, there exist* $\overline{u} \in \mathcal{V}er(\mathrm{L})$ *and* $\overline{v} \in \mathcal{V}er(\mathrm{G})$, *and a unifying extension* $\langle E, \zeta\rangle$ *of the equation* $\mathrm{L}[\overline{u}] = \mathrm{G}[\overline{v}]$ *such that* $U = \mathrm{L} \otimes_\zeta E = \mathrm{G} \otimes_\zeta E$.

Proof. Let A be the subset of internal vertices of U which are also internal vertices of L and of G, roots of $\mathrm{L}, U, \mathrm{G}$ being considered as specific internal vertices. By assumption, $A \neq \varnothing$, implying that L and G overlap. The core of the proof is the definition of two lists $\overline{v}, \overline{w}$ of partner vertices of L, G which generate A, that is, all vertices of A are accessible from $\overline{v}$ in L and from $\overline{w}$ in G. As partner vertices, v_i and w_i must coincide, that is, be identified in A. Let us denote vertices of A by u, v and w according to their origin, in U, L and G respectively.

Because left-hand sides of rules are clean drags, for all internal vertices $u \in A$, there exists some root $r \in \mathcal{R}(\mathrm{L}) \cup \mathcal{R}(\mathrm{G})$ such that, $r\,(X^*_{\mathrm{L}} \cup X^*_{\mathrm{G}})\,v$. There are therefore three kinds of partner vertices (v, w): v, w are roots of L, G; or v is a root of L and w is not a root of G; or v is not a root of L and w is a root of G. Eliminating redundancies (with respect to accessibility) yields two lists of vertices that satisfy the conditions for being partner vertices, and generate A.

We now show that U defines a unifying extension of the equation $\mathrm{L}[\overline{v}] = \mathrm{G}[\overline{w}]$. Since L and G match U, $U = \mathrm{L}[\overline{v}] \otimes_\gamma C = \mathrm{G}[\overline{w}] \otimes_\delta D$ for some rewriting extensions $\langle C, \gamma\rangle$ and $\langle D, \delta\rangle$ of L and G respectively. Assuming that L, G are renamed apart, then $U = (\mathrm{L}[\overline{v}] \oplus \mathrm{G}[\overline{w}]) \otimes_{\gamma\cup\delta} (C \oplus D)$. Hence L and G are unifiable at partner vertices $(\overline{v}, \overline{w})$ with the extension $(C \oplus D, \gamma \cup \delta)$. ◂

▸ **Definition 18** (Critical pair). Let $\mathrm{L} \to \mathrm{R}$ and $\mathrm{G} \to \mathrm{D}$ be two rules that are unifiable at partner vertices $\overline{v}, \overline{w}$, and $\langle C, \xi\rangle$ be an mgu. Then, $\langle \mathrm{L} \otimes_\xi C, \mathrm{G} \otimes_\xi C\rangle$ is called a *critical pair* of $\mathrm{L}, \mathrm{G} \to \mathrm{D}$ at $\overline{v}, \overline{w}$.

Note that the notion of drag critical pair becomes symmetric.

Given a drag rewriting system, how many critical pairs can be generated ? Their number is indeed bounded by the potential choices for partner vertices, which must satisfy the constraints stated in the proof of Lemma 17. In practice, this number should remain small, as is the case for terms.

We can now end up with our main result, which follows easily from Lemmas 15, 16 and 17:

▸ **Theorem 19.** *Let* $\mathcal{S}$ *be a terminating rewrite system on drags. Then,* $\mathcal{S}$ *is confluent iff all its critical pairs are joinable.*

As for terms: (i) termination is not essential: the same result could be rephrased, we believe, by using decreasing diagrams instead of joinability; and (ii) redundancy criteria [1], should allow to filter out useless critical pairs.

4 Conclusion

Drags appear to be an extremely handy generalization of terms, dags and jungles: the intuitions behind them all are very similar, as well as the most important algorithms for implementing rewriting and testing its termination and confluence, despite the possibility of having arbitrary cycles in drags. This is made possible by a powerful composition operator.

Drags do not exactly generalize terms, though, as is pointed out in [5]. This is because our definition of composition *forces* sharing, as does term rewriting in practice. Capturing the term case requires using a composition operator based on drag isomorphism instead of drag equality in presence of non-linear variables in rules. This is of course possible, and is currently being investigated.

Finally, the present result together with the drag path ordering given in [4] shoud allow the development of completion procedures for drag rewriting systems.

Warm thanks to Anne Yenan and José Motos who provided a *deluxe roof* to the first author during his one month stay in Barcelona at the invitation of the second author.

References

1 Leo Bachmair and Harald Ganzinger. On Restrictions of Ordered Paramodulation with Simplification. In Mark E. Stickel editor, *10th International Conference on Automated Deduction*, Kaiserslautern, July 24-27, 1990. Lecture Notes in Computer Science 449, pages 427–441, Springer, 1990.

2 Horatiu Cirstea and David Sabel, editors. *Proceedings Fourth International Workshop on Rewriting Techniques for Program Transformations and Evaluation, WPTE@FSCD 2017, Oxford, UK, September 2017*, volume 265 of *EPTCS*, 2018.

3 Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 243–320. Elsevier, 1990.

4 Nachum Dershowitz and Jean-Pierre Jouannaud. Graph path orderings. In Gilles Barthe, Geoff Sutcliffe, and Margus Veanes, editors, *LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 57 of *EPiC Series in Computing*, pages 307–325. EasyChair, 2018.

5 Nachum Dershowitz and Jean-Pierre Jouannaud. Drags: A compositional algebraic framework for graph rewriting. *Theor. Comput. Sci.*, 777:204–231, 2019.

6 Annegret Habel, Hans-Jörg Kreowski, and Detlef Plump. Jungle evaluation. *Fundam. Inform.*, 15(1):37–60, 1991.

7 Gérard Huet. *Unification dans les langages d'ordre 1, ...,ω*. PhD thesis, Université Paris 7, Paris, France, 1976.

8 Jean-Pierre Jouannaud and Fernando Orejas. Unification of drags. In *UNIF 2020*. available at https://hal.inria.fr/hal-02562152.

9 Donald Knuth and Peter Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

10 Vincent van Oostrom. Confluence by decreasing diagrams. *Theor. Comput. Sci.*, 126(2):259–280, 1994.