Product Based Graph Rewriting and its Confluence

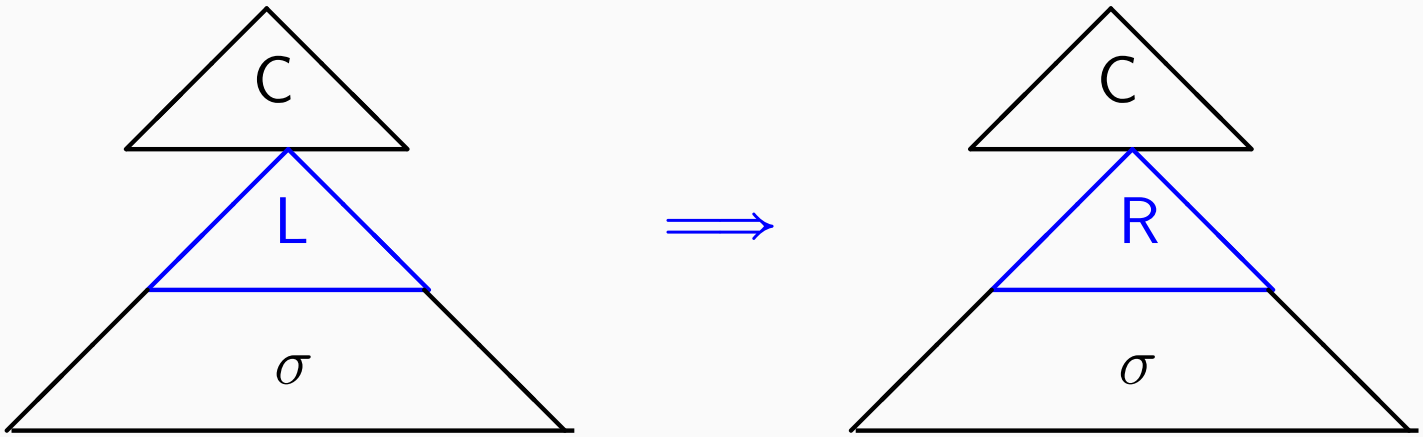
Jean-Pierre Jouannaud

Joint work with
Nachum Dershowitz and **Fernando Orejas**

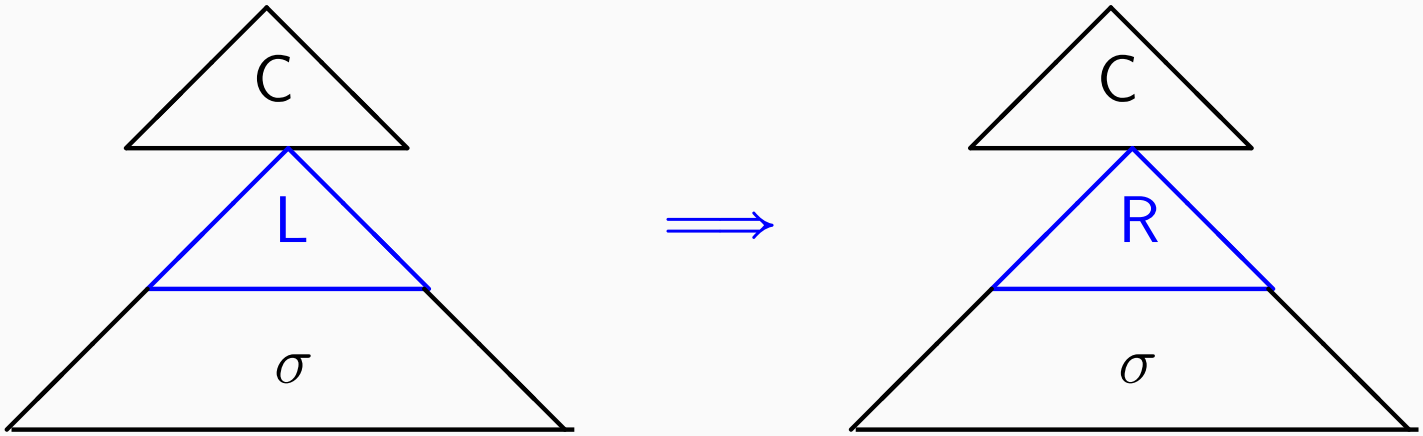
July 17, 2026

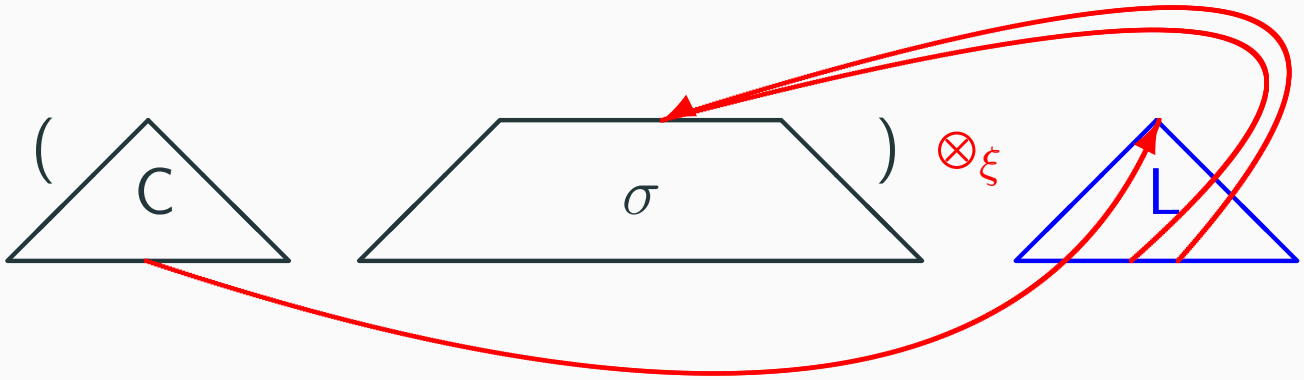
Scale term rewriting theory to graphs

Tree Rewriting and Composition

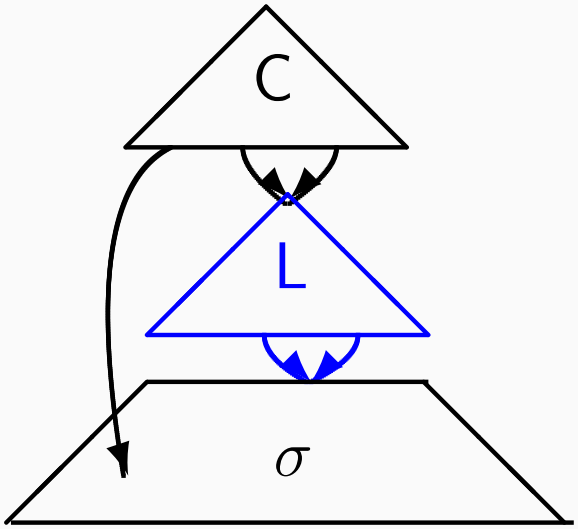


Tree Rewriting and Composition

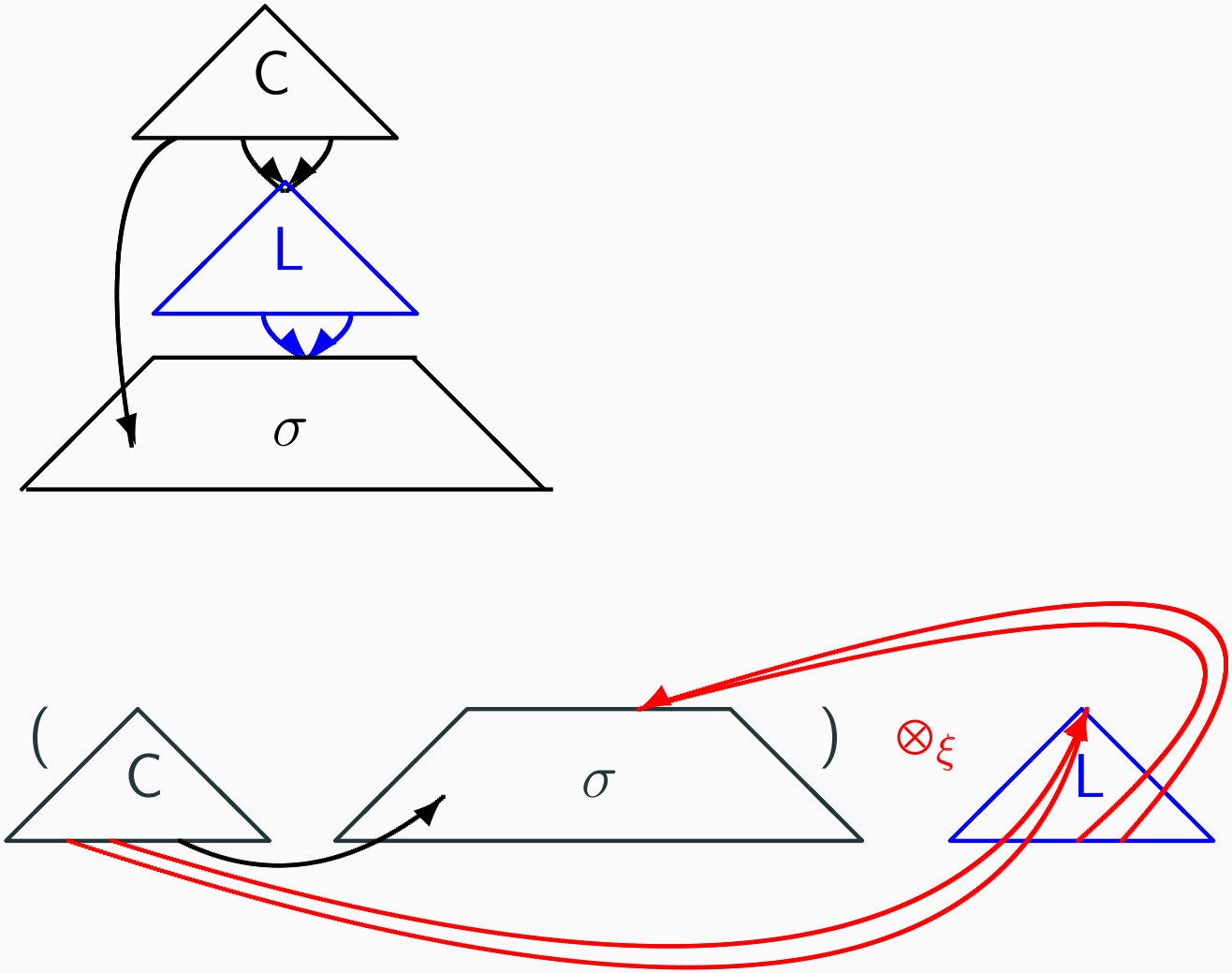




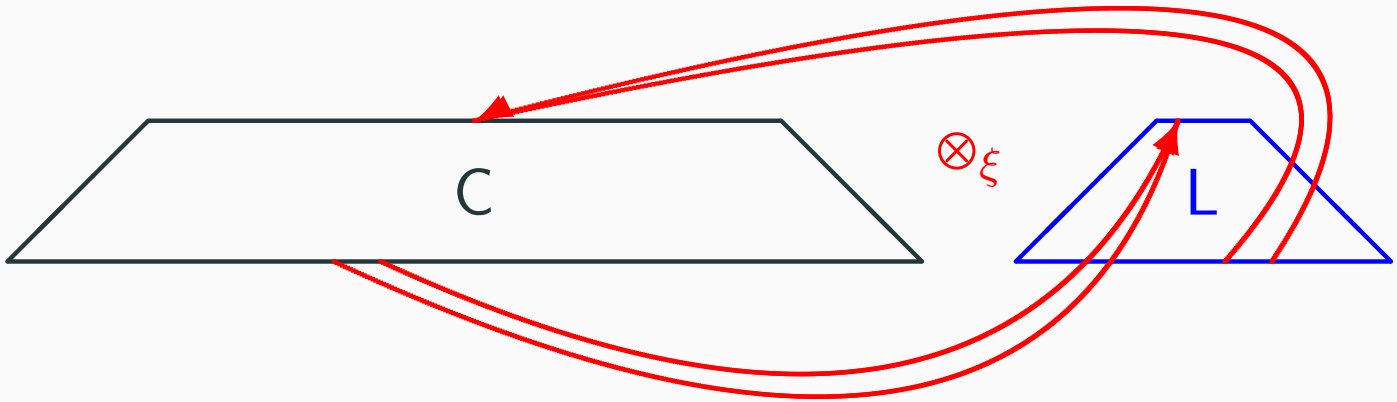
Dag Rewriting and Composition



Dag Rewriting and Composition



Graph Rewriting and Composition



We need multi-rooted multi-graphs with variables labeling some leaves

Graphs

Rooted Graphs

Σ function symbols labeling vertices: 

Ξ variable symbols with arity zero: 

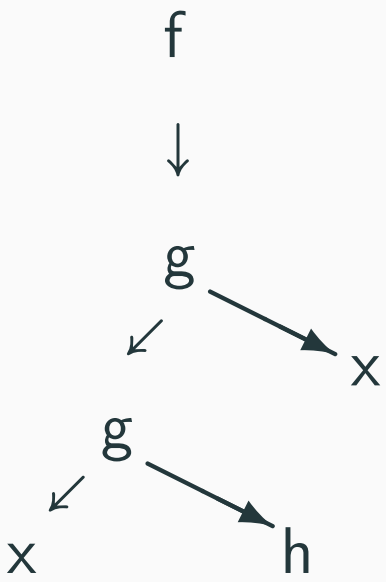
A *compositional graph* consists of

- *vertices* V
- *sprouts* $S \subseteq V$, leaving $I = V \setminus S$ for the *internal vertices*
- *edges* E is a set, an edge is a triple $e : u \to v$, $u \notin S$, name e, tail u, head v, restrictions E_I and E_S depending on v
- *roots* R is a set; $R(v)$ is the set of roots pointing at $v \in V$
- *labels* $L : S \to \Xi$
  $L : I \to \Sigma$

Ground compositional graphs have neither roots nor sprouts

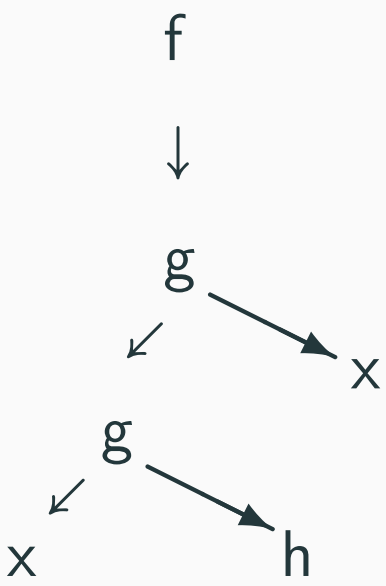
Labels in Σ, possibly indexed, will serve as names for vertices

Example: Tree, Dag, Drag, Graph

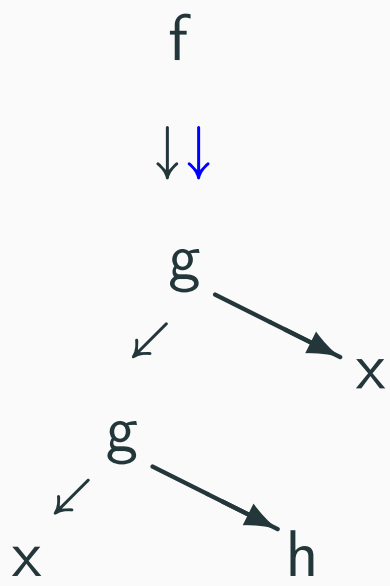


Term

Example: Tree, Dag, Drag, Graph

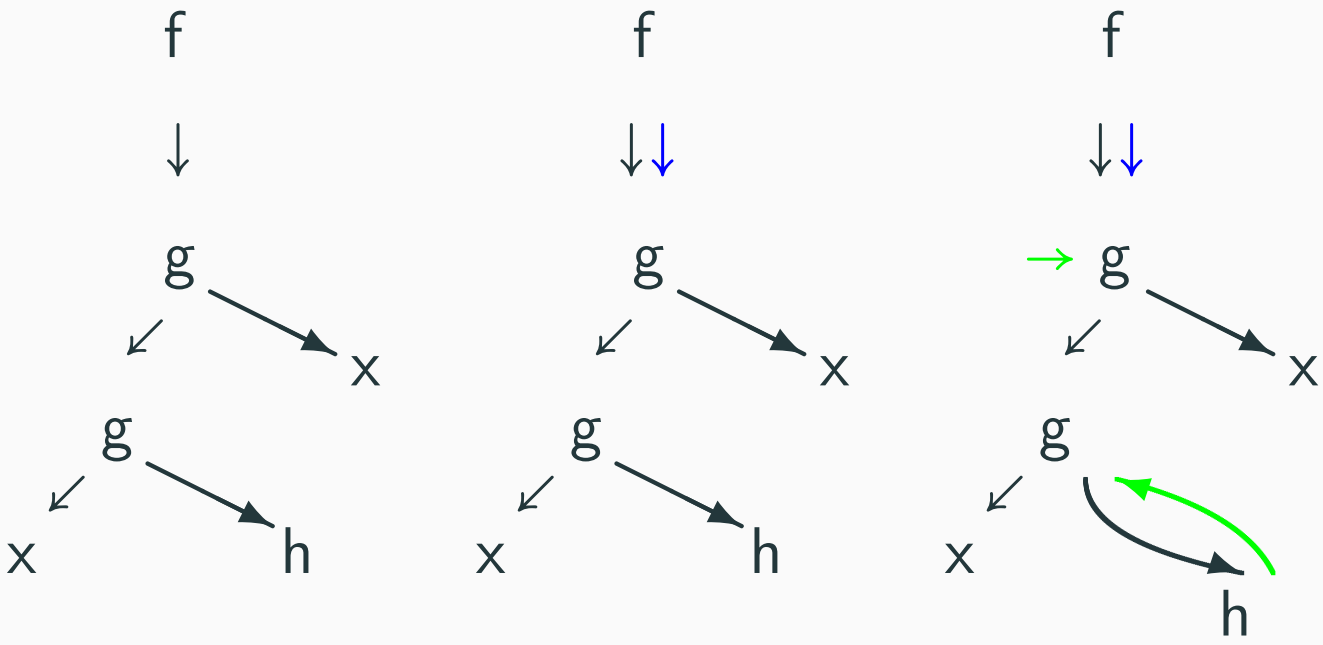


Term



Dag

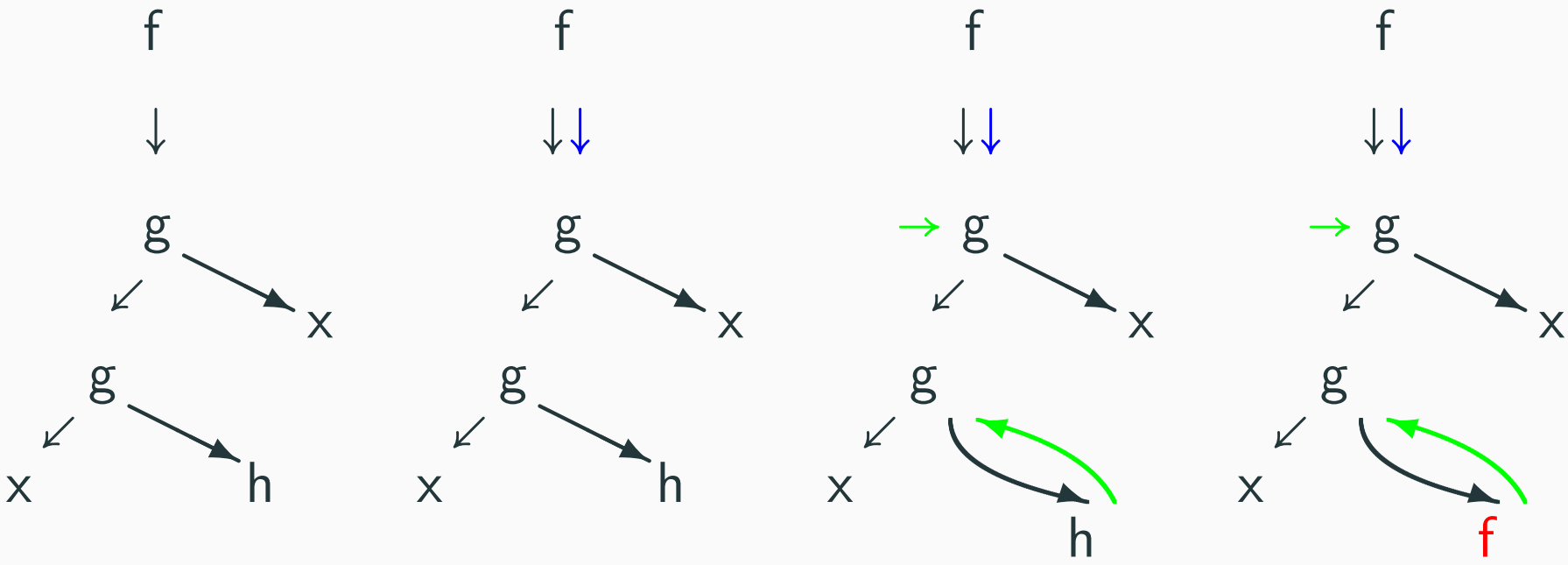
Example: Tree, Dag, Drag, Graph



Term Dag Drag

Term Graph

Example: Tree, Dag, Drag, Graph



Term

Dag

Drag

Term Graph

Compositional

Graph

Morphisms on compositional graphs

Miscalleneous

– *Subgraphs:* generated by a set of vertices w: $D|_W$

– *Connected component:* set of vertices closed under successors and predecessors, and equal labeling of sprouts

– *Equimorphic graphs:* identical up the name of their vertices

– *Compacted graphs:* no two subgraphs are equimorphic

– *Sharing equivalent graphs:* equimorphic when compacted

Morphisms

Morphism $o : D = \langle V, S, E, R, L \rangle \hookrightarrow D' = \langle V', S', E', R', L' \rangle$

– $o_{\mathcal{V}} : V \to V'$, the *vertex map*

– $o_{\mathcal{E}} : E \to E'$, the *edge map*

– $o_{\mathcal{R}} : R \to R'$, the *root to root map*

– $o_{\mathcal{R},\mathcal{E}} : R \to E'$, the *root to edges map*

1. $o_{\mathcal{V}}$ restricts to mappings from I to I', and preserves labels;
2. $o_{\mathcal{V}}$ restricts to mappings from isolated sprouts to isolated sprouts;
3. $o_{\mathcal{E}}, o_{\mathcal{R}}, o_{\mathcal{R}\to\mathcal{E}}$ commute with $o_{\mathcal{V}}$
4. o preserves sharing equivalent subgraphs

Notation: $o_{\mathcal{E}_I}, o_{\mathcal{E}_S}$ restrictions of $o_{\mathcal{E}}$ to internal- and demi-edges

Matching morphims

Matching morphism $o : D \hookrightarrow D'$ is a morphism such that

– $o_{\mathcal{V}}$ restricts injectively to I and isolated sprouts

– $o_{\mathcal{E}}$, $o_{\mathcal{R},\mathcal{R}}$ and $o_{\mathcal{R},\mathcal{E}}$ are injective

– $\mathcal{I}m(o_{\mathcal{E}_S}) \subseteq \mathcal{I}m(o_{\mathcal{R},\mathcal{E}})$

We say that D is *embedded* into D' and write $o : D \hookrightarrow D'$.

An *injection* is a matching morphism whose vertex map is the identity on internal vertices and isolated sprouts. An *isomorphism* is a matching morphism whose vertex, edge, and root maps are all bijective ($o_{\mathcal{R},\mathcal{E}}$ being then empty).

Operations on compositional graphs

Parallel and cyclic compositions

Two graphs D, D' are *compatible* if

- shared internal vertices share their label
- if some variable x is shared, some shared sprout is labeled x
- shared edges share their sources and their targets
- shared roots share their vertex

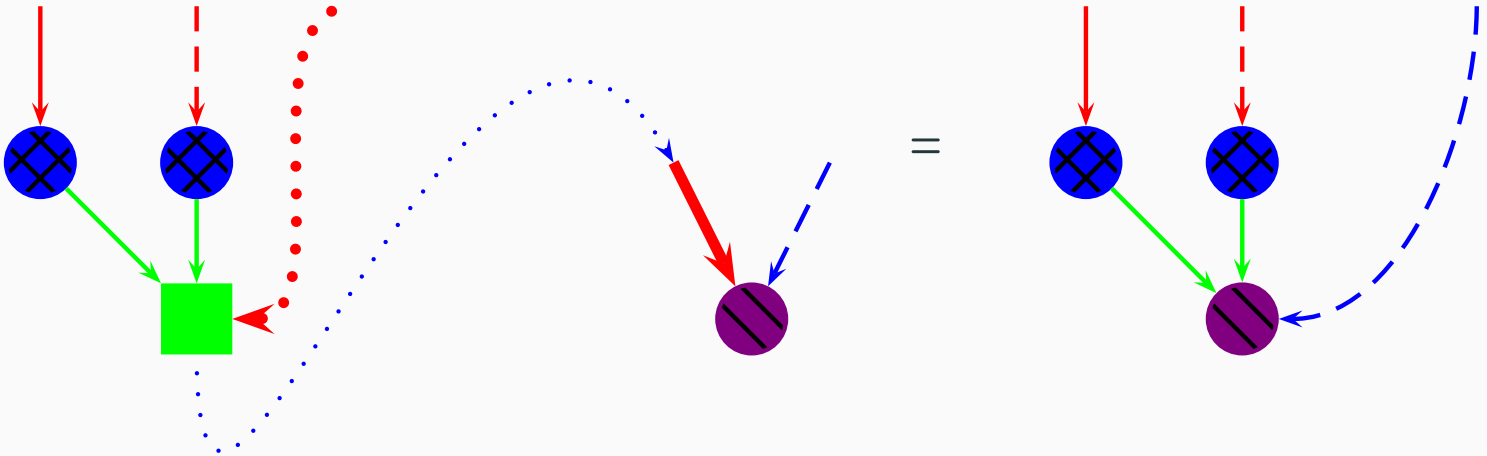
Parallel and cyclic compositions

Two graphs D, D' are *compatible* if
- shared internal vertices share their label
- if some variable x is shared, some shared sprout is labeled x
- shared edges share their sources and their targets
- shared roots share their vertex

If D, D' are compatible,

- their parallel composition is $D \oplus D'$
- their cyclic composition wrt switchboard ξ is $D \otimes_\xi D'$
- cyclic composition is obtained by wiring $D \oplus D'$ wrt ξ, where ξ tells you where to branch sprouts from one side to roots of the other side

Wiring



A wire $s \rightsquigarrow u$ is a pair made of a sprout s and an vertex $u \neq s$

A switchboard ξ for D, D' is a set of wires $(\xi_{D \to D'}, \xi_{D' \to D})$ s.t. equally labeled sprouts are replaced by vertices yielding sharing equivalent subgraphs in the wired graph

Wiring eats as many roots of the wire's target as the number of predecessors of the wire's origin, whose roots are lost.

Major properties

Algebra of rooted graphs: parallel and cyclic compositions are associative, commutative, and have the empty graph as neutral element.

Major properties

Algebra of rooted graphs: parallel and cyclic compositions are associative, commutative, and have the empty graph as neutral element.

Matching Theorem: Assume $o : L \hookrightarrow G$ bis a matching morphism such that L is a *pattern*, that is, a *graph without isolated sprout*. Then, there exists a *rewriting extension* (C, ξ)

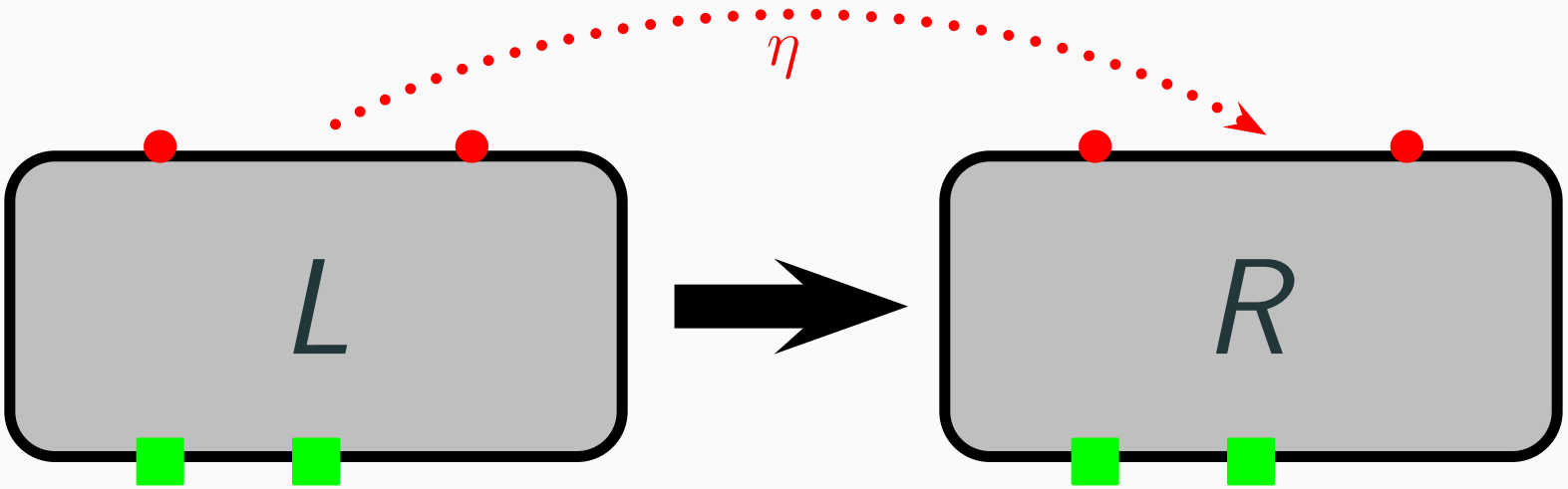
- *C is a linear compositional graph*
- *ξ_C is surjective on the roots of L*
- *ξ_L is total*

such that

$$G = C \otimes_{\xi} L$$

Product Based Rewriting

Rules as (pairs of) drags



- L, R are drags, the *pattern* L contains no isolated sprout
- all variables have an occurrence in the pattern
- *root-map* $\eta : \mathcal{V}_L \to \mathcal{V}_R$ *preserves roots in number*
- *product based rewriting:* $C \otimes_\xi L \longrightarrow C' \otimes_{\xi'} R$
- $(C, \xi), (C', \xi')$ are *rewriting extension* of L and R s.t. C, C' are sharing-equivalent and share no vertex with L, ξ, ξ' are sharing equivalent sets of wirings
- root-map η ensures the absence of dangling edges

Drags: Current research status

Results

- Term rewriting is a particular case of drag rewriting
- Termination: a recursive path ordering for drags (can be total on ground drags)
- Unification: a square complexity propagation algorithm for drags by marking pairs of corresponding vertices
- Local confluence for drags by joinability of critical pairs

Results

- Term rewriting is a particular case of drag rewriting
- Termination: a recursive path ordering for drags (can be total on ground drags)
- Unification: a square complexity propagation algorithm for drags by marking pairs of corresponding vertices
- Local confluence for drags by joinability of critical pairs

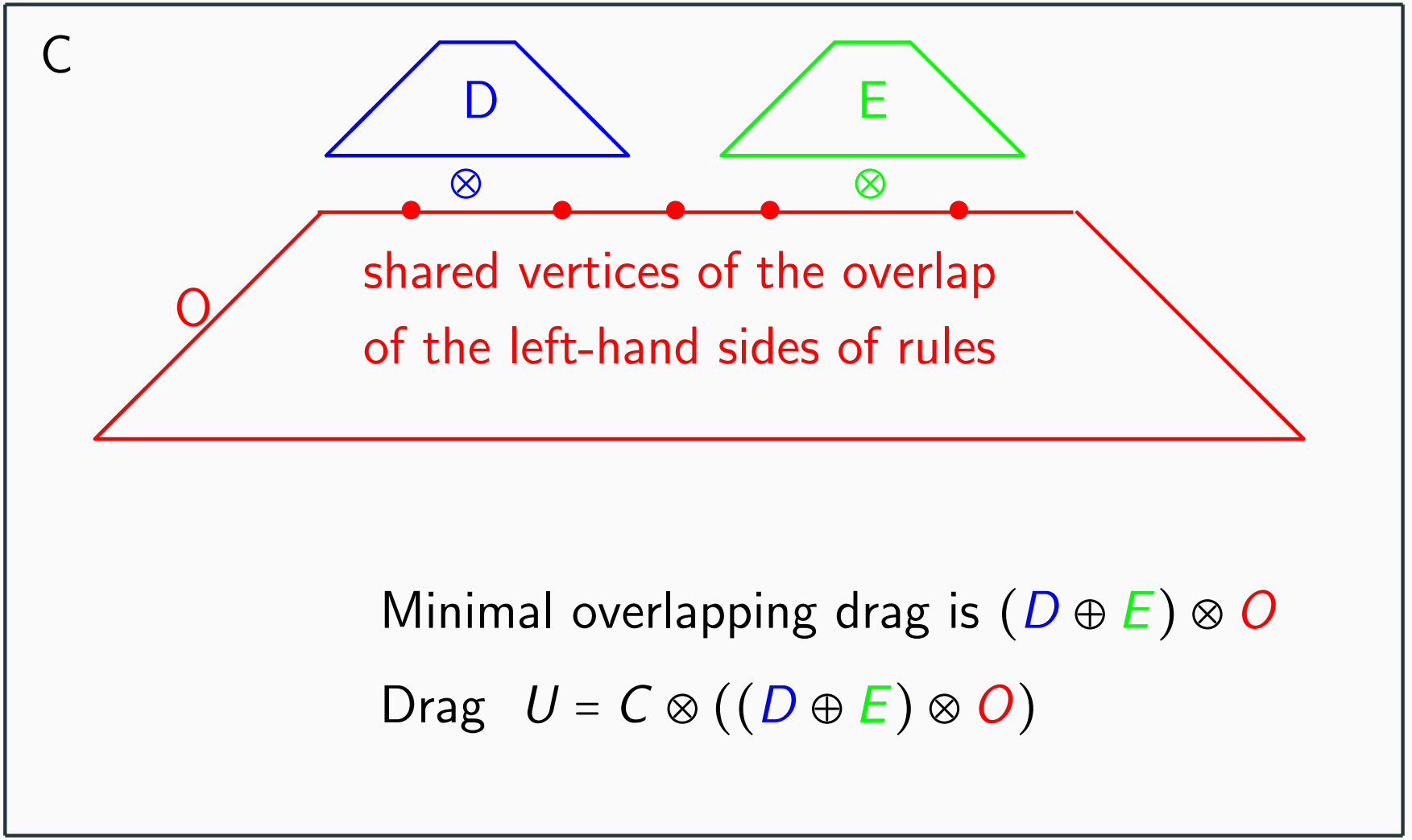
  Proof sketch: let U rewrite with rules $L \to R$ and $G \to D$.
  - overlapping case uses drag unification for computing critical pairs, and lifting their joinability by monotonicity
  - non-overlapping case thanks to AC-proprerty of $\otimes$:

$$U = C \otimes (L \oplus G) = C \otimes (L \otimes G)$$

$$U = (C \otimes L) \otimes G \longrightarrow (C \otimes L) \otimes D = (C \otimes D) \otimes L \longrightarrow (C \otimes D) \otimes R = V$$

$$U = (C \otimes G) \otimes L \longrightarrow (C \otimes G) \otimes R = (C \otimes R) \otimes G \longrightarrow (C \otimes R) \otimes D = V$$

Overlapping case



Minimal overlapping drag is $(D \oplus E) \otimes O$

Drag $U = C \otimes ((D \oplus E) \otimes O)$

Unification

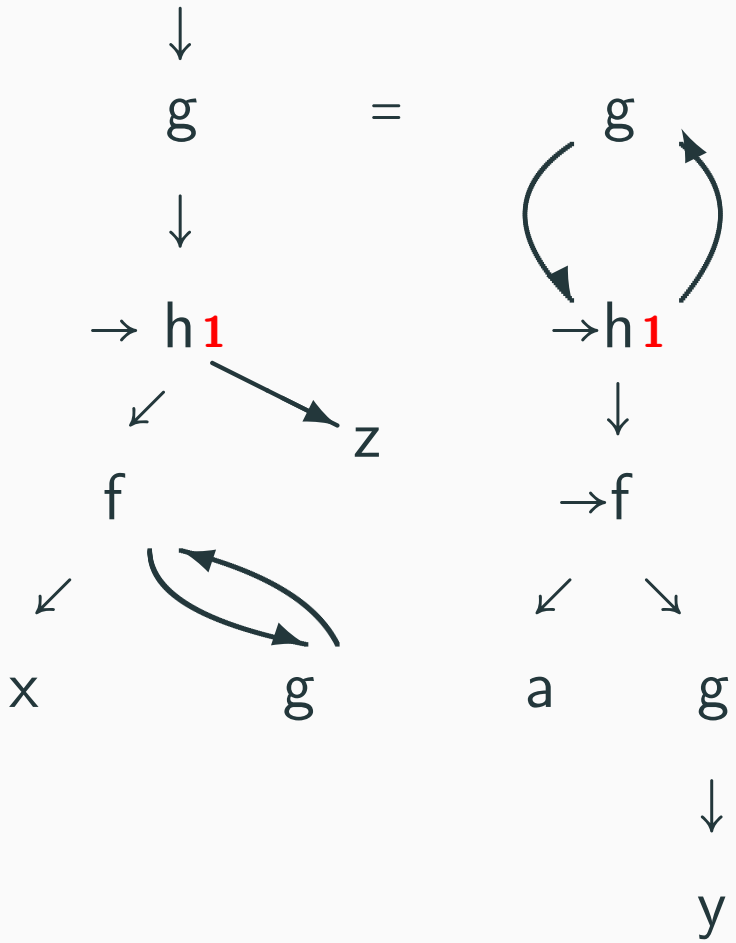
Drag unification

Given drags U, V, we call partner vertices two lists L_U, L_V of equal length of internal vertices of U and V, such that no two vertices $u, u' \in L_U$ (resp., $v, v' \in L_V$) are related by a path.

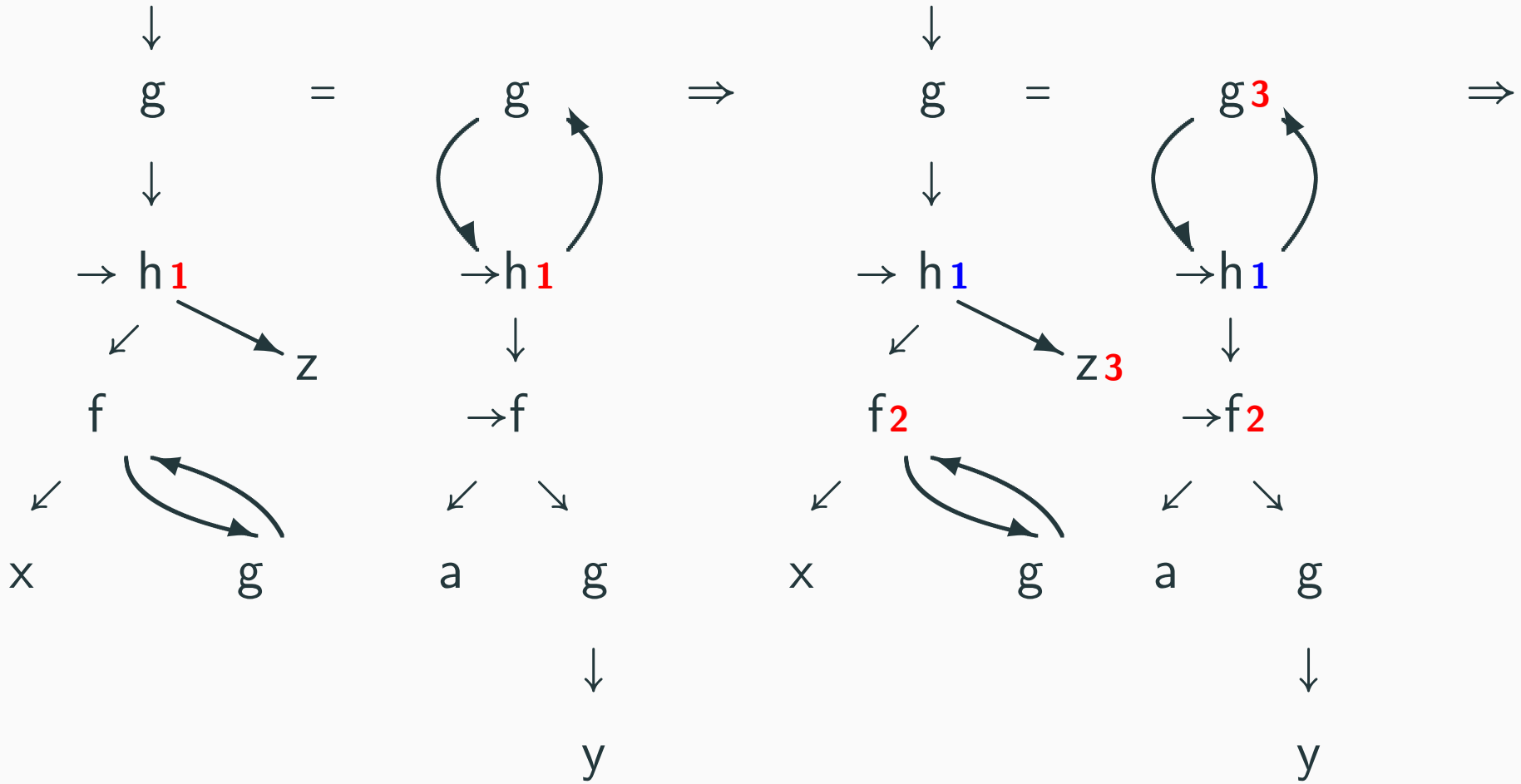
A drag unification problem, $U[\overline{u}] = V[\overline{v}]$, is a pair (U, V) of connected patterns that have been renamed apart, together with partner vertices $P = (\overline{u}, \overline{v})$. A solution (or *unifier*) of the drag unification problem $U[\overline{u}] = V[\overline{v}]$ is a pair of rewriting extensions (C, ξ) and (D, ζ) of U and V respectively, such that $C \otimes_\xi U$ and $D \otimes_\zeta V$ are *identified at* $(\overline{u}, \overline{v})$.

A drag equal modulo renaming to $C \otimes_\xi U$ (hence to $D \otimes_\zeta V$) is called overlap of U, V at $(\overline{u}, \overline{v})$.

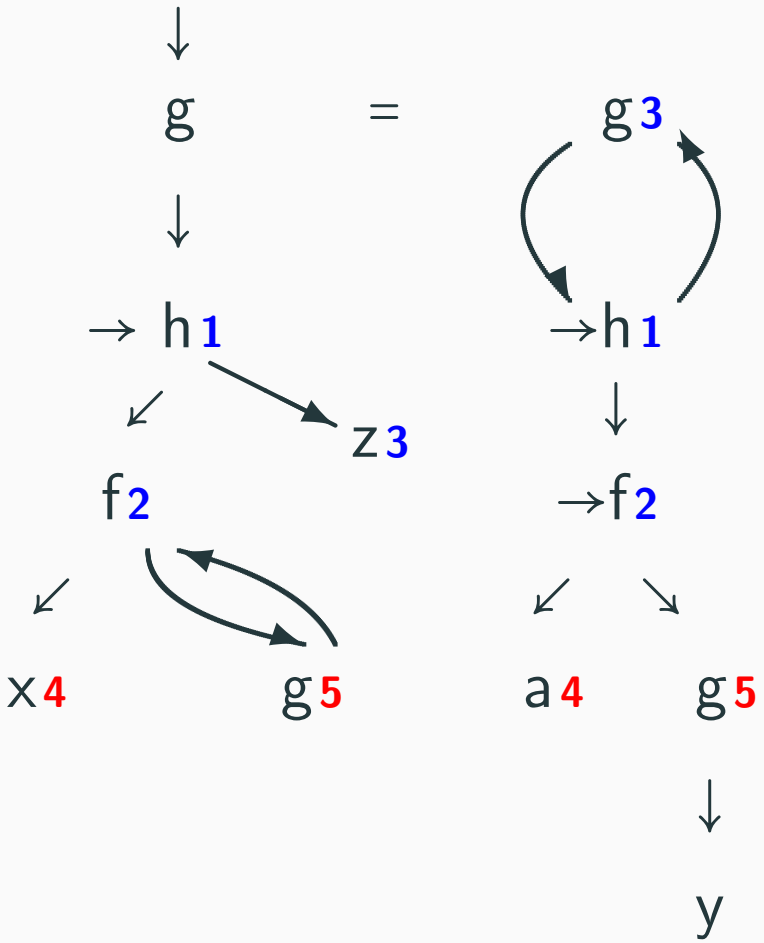
Drag unification rules at work (given L_U, L_V) 1/3



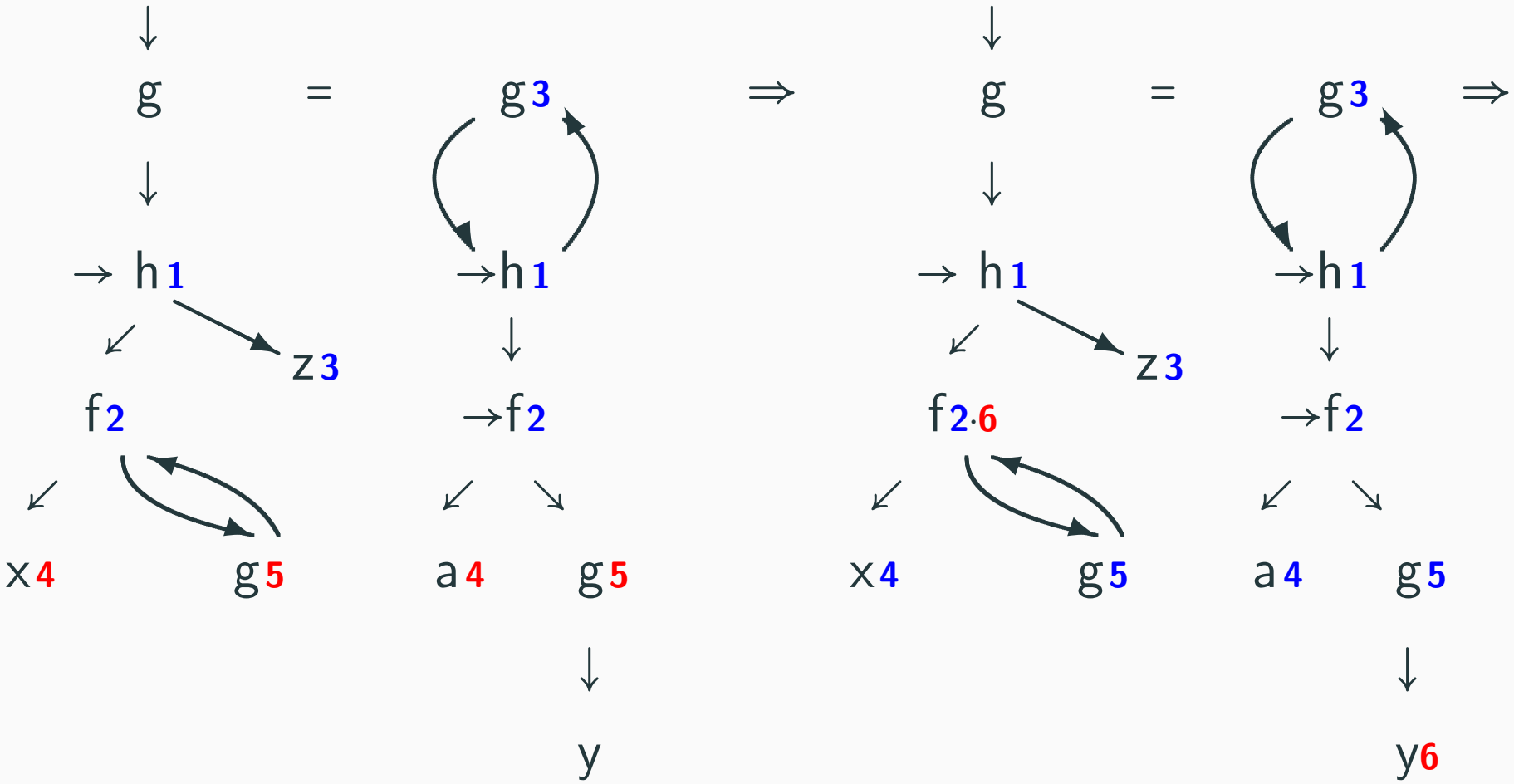
Drag unification rules at work (given L_U, L_V) 1/3



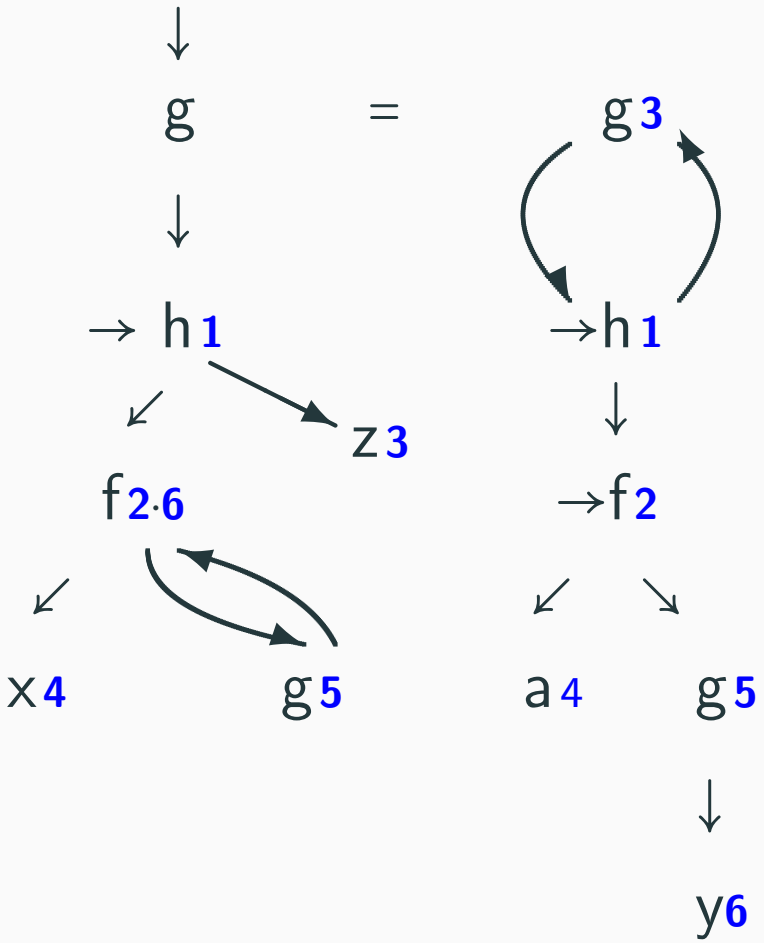
Drag unification rules at work 2/3



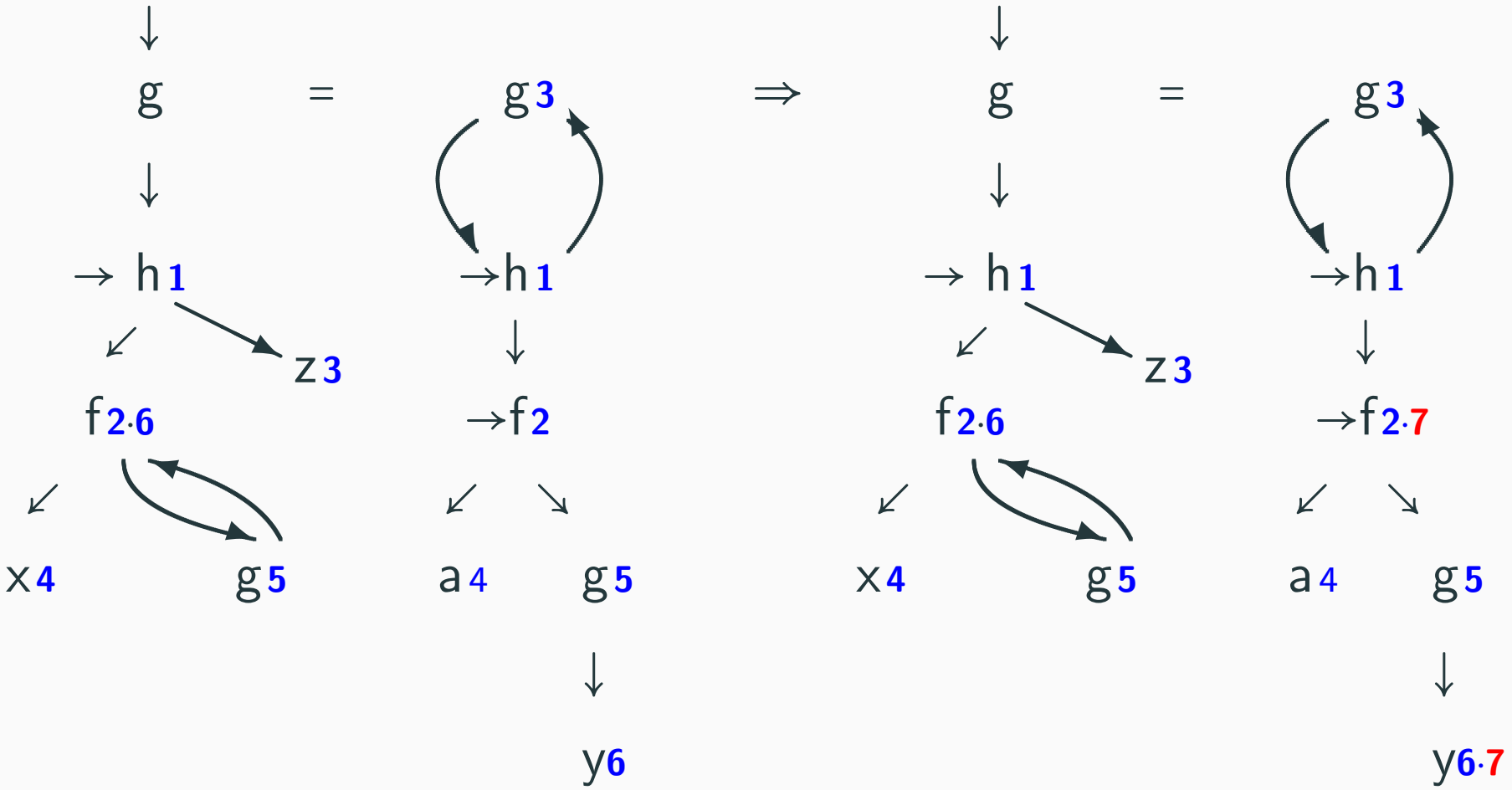
Drag unification rules at work 2/3



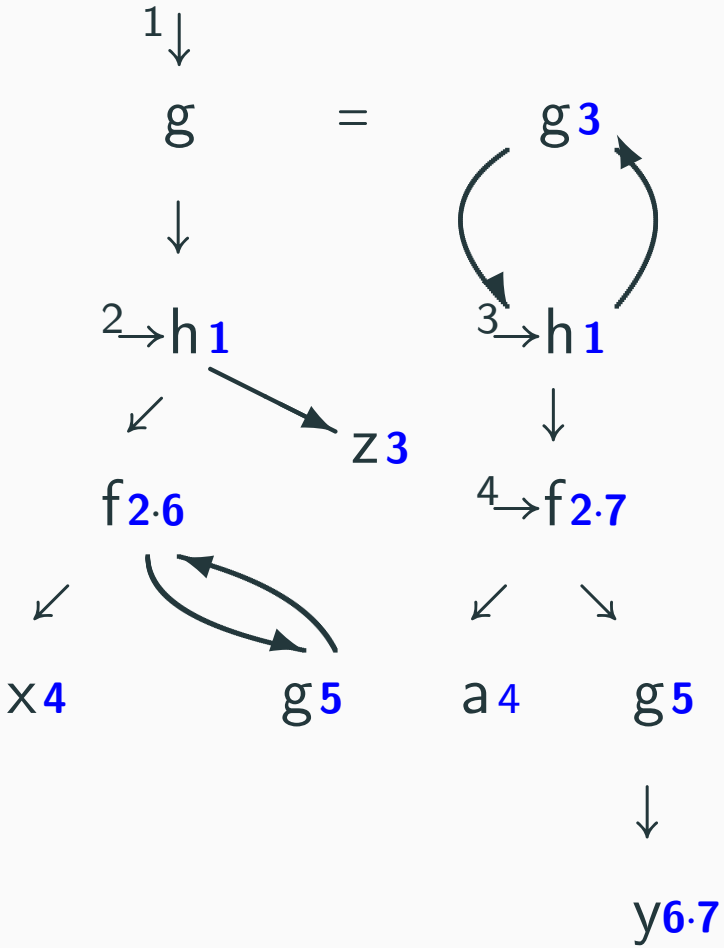
Drag unification rules at work 3/3



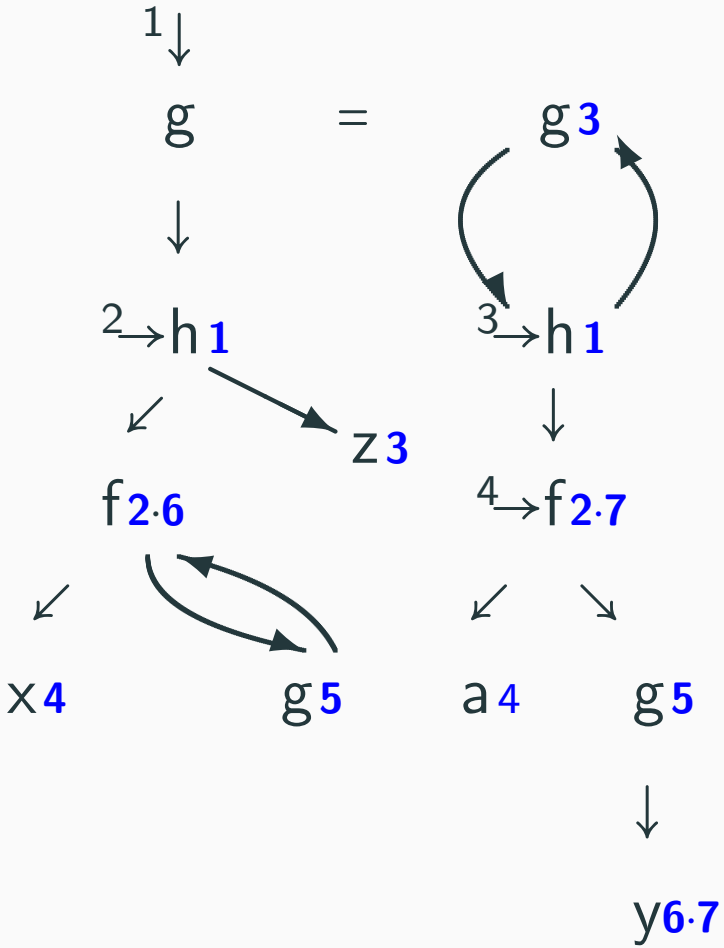
Drag unification rules at work 3/3



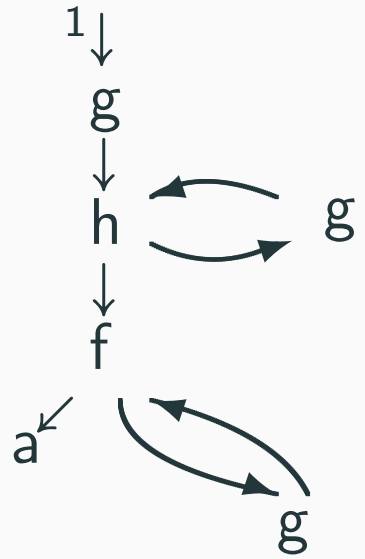
Most general unifier



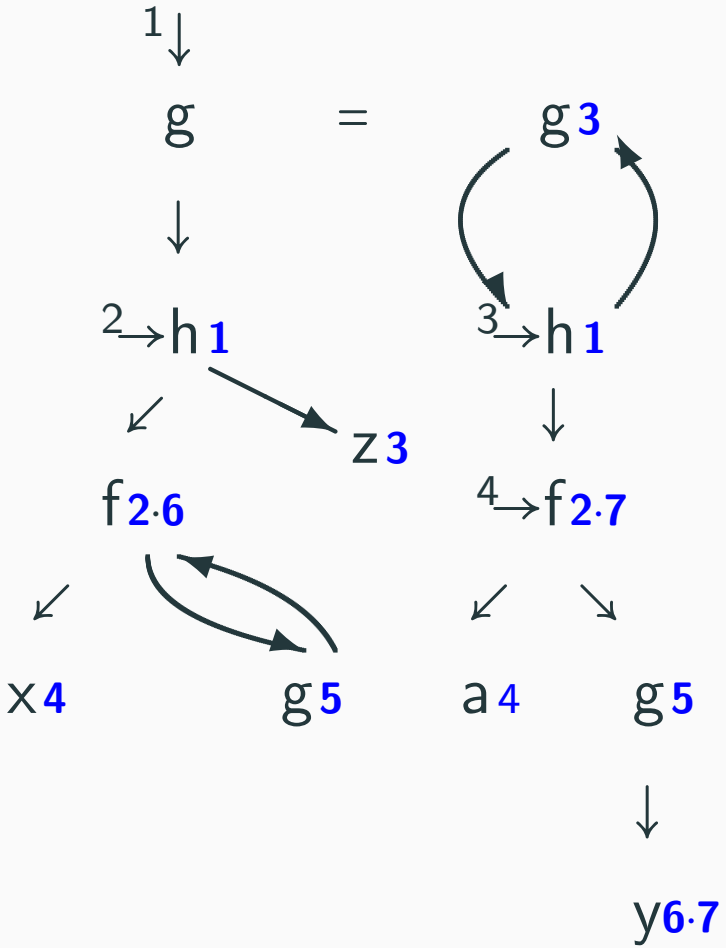
Most general unifier



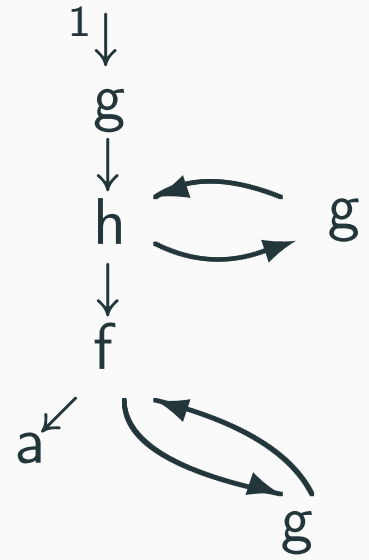
Overlap



Most general unifier



Overlap



$C = g(\overset{5}{\downarrow} a, \overset{6}{\downarrow} z')$ $\quad \xi = \{x \mapsto 6, z \mapsto 5, z' \mapsto 2\}$

$D = g(\overset{1}{\downarrow} y', \overset{7}{\downarrow} x')$ $\quad \zeta = \{y \mapsto 7, y' \mapsto 4, x' \mapsto 3\}$

Local confluence of compositional graphs

Step 1: reduce (non-deterministically) unification of compositional graphs to drag unification

Step 2: apply local confluence of drags

Step 3: deduce local confluence of compositional graphs

Drag rewriting versus Term rewriting

Drag rewriting encompasses Term rewriting

Given a term t, let $[t]$ be the drag obtained by adding a root at the head of t, and $[R] = \{[l_i] \to [r_i]\}_i$ if $R = \{l_i \to r_i\}_i$.

Theorem: $t \xrightarrow[R]{} t'$ implies $[t] \xrightarrow[[R]]{} [t']$. Conversely, $[t] \xrightarrow[[R]]{} t''$ and t'' is the encoding of a term, implies that $t \xrightarrow[R]{} t'$ where $[t']$ and t'' are equimorphic.

Were t'' not assumed to the encoding of a term, then $[t']$ and t'' would only be sharing equivalent.

Property: Drag unification of term encodings reduces to term unification.

Because term encodings don't have roots at strict subterms.

Consequence: local confluence is preserved by the encoding.

DPO versus PBR

DPO

Rule: $L \hookleftarrow C \hookrightarrow R$ (This is called a *span*)

Match: $L \hookrightarrow G$

Pushout complement: $G \hookleftarrow C' \hookleftarrow C$

Pushout: $C' \hookrightarrow G' \hookleftarrow R$

This constructrion requires the existence of pushouts and pushout complements.

It works in any adhesive category of graphs with pushouts.

PBR compared with DPO

- *No explicit variables in DPO, explicit variables in PBR*
- *Implicit linear variables via monomorphisms in DPO*
- *DPO: no known way to encode non-linear rewriting*
- *DPO does NOT generalize term rewriting, PBR does*
- *DPO may generate dangling (a source, no target) edges*
- *DPO works uniformly over adhesive categories of graphs*
- *Adhesivity must be checked for each category of graphs*
- *Product defined uniformly for all kinds of rooted graphs*
- *Matching Theorem proved for each category of graphs*
- *The category of ground compositional graphs is adhesive*
- *The category of compositional graphs is not adhesive*
- *DPO: Confluence is undecidable for terminating systems*
- *PBR: Confluence decidable for terminating systems*

Bibliography

- Nachum Dershowitz, Jean-Pierre Jouannaud
  Graph Path Orderings. LPAR 2018: 307-325
- Nachum Dershowitz, Jean-Pierre Jouannaud
  Drags: A compositional algebraic framework for graph rewriting. Theor. Comput. Sci. 777: 204-231 (2019)
- Jean-Pierre Jouannaud, Fernando Orejas
  Unification of Drags and Confluence of Drag Rewriting. J. of Log. and Alg. Methods in Programming 131 (2023)
- Nachum Dershowitz, Jean-Pierre Jouannaud, Fernando Orejas, Drag Rewriting, In Yuri Gurevich's Festschrift, Guillermo Badia Ed., to appear in October 2026
- Nachum Dershowitz, Jean-Pierre Jouannaud, Fernando Orejas, Graph Rewriting, working paper