# Semantics-Preserving DPO-Based Term Graph Rewriting*

**(Extended Abstract)**

Wolfram Kahl
McMaster University, Hamilton, Ontario, Canada,
kahl@cas.mcmaster.ca

Yuhang Zhao
McMaster University, Hamilton, Ontario, Canada,
zhaoy36@mcmaster.ca

Term graph rewriting is important as "conceptual implementation" of the execution of functional programs, and of data-flow optimisations in compilers. One way to define term graph transformation rule application is via the well-established and intuitively accessible double-pushout (DPO) approach; we present a new result proving semantics preservation for such DPO-based term graph rewriting.

## 1 Introduction and Related Work

Term graph rewriting goes back to Wadsworth (1971), who proposed it as an efficient implementation mechanism for the $\lambda$-calculus. This aspect has remained dominant in the term graph literature; for example, Rose (1993) defines an operational semantics of a lazy functional programming language via term graph rewriting; Ariola et al. (2000) study confluence of term graph rewriting using bisimilarity.

When justifying term graph rewriting as a correct implementation technique (for, in particular, functional programming), most of the literature approaches this from the relationship with term rewriting. For example, when Plump (2002) writes about "Essentials of Term Graph Rewriting", soundness and completeness are considered only with respect to term rewriting. Kennaway et al. (1993, 1994) define a notion of simulation to prove adequacy of term graph rewriting for finite and rational term rewriting.

When attempting to employ traditional categorial approaches to graph rewriting, the so-called "algebraic approach", to term graph rewriting, two main problems arise: First, categories of "standard" term graph homomorphisms typically do not have all pushouts, since unification translates into pushouts, and second, the interface graphs needed both for the double-pushout (DPO) approach and for the single-pushout approach (to capture the domain of morphisms) are typically not term graphs, but some kind of "term graphs with holes". Term graph rewriting is therefore a niche of graph transformation that has pioneered exploration of formalisms where pushout squares are generalised in some way, in particular by using different morphisms in the horizontal and vertical directions of the standard DPO drawing.

For example, Banach (1993) defines DACTL term graph rewriting using a modified opfibration, and Kahl (1996, 1997) uses both fibrations and opfibrations to define rewriting of term graphs with variable-binding constructs. A different approach to using separate classes of horizontal and vertical morphisms for term graph rewriting has been proposed by Duval et al. (2009), who are using a specific rule concept as morphisms in the horizontal direction in their "heterogeneous pushout approach". (More recently, motivated by attributed graphs, which share some characteristics with term graphs, Habel and Plump (2012) propose "$\mathcal{M},\mathcal{N}$-adhesive transformation systems" as one general framework to accommodate different classes of morphisms in the horizontal and vertical directions of the double-pushout setting.)

*This research is supported by the National Science and Engineering Research Council of Canada, NSERC.

Submitted to:
TERMGRAPH 2018

Corradini and Gadducci (1999a, 2002) opened up a new way of investigating term graphs by defining gs-monoidal categories as a variant of Lawvere theories (Lawvere, 1963); they showed that taking natural numbers as objects and term graphs with $m$ inputs and $n$ outputs as morphisms from object $m$ to object $n$ produces a free gs-monoidal category, and thus automatically obtained a functorial semantics for term graphs in arbitrary gs-monoidal categories, which include all Cartesian categories, and so in particular also *Set*. Continuing this line of work, Corradini and Gadducci (1997, 1999b) obtain semantics preservation for a low-level definition of "ranked dag rewriting", and involving "contexts" analogous to the contexts of term rewriting. Finally, Corradini and Gadducci (2005) show a quasi-adhesive category of term graphs, but emphasise that adhesive categorial rewriting in that category does not quite match term graph rewriting. They mention in their conclusion that a possible alternative is to perform the DPO on a super-category of hypergraphs; this is essentially the approach we are elaborating here.

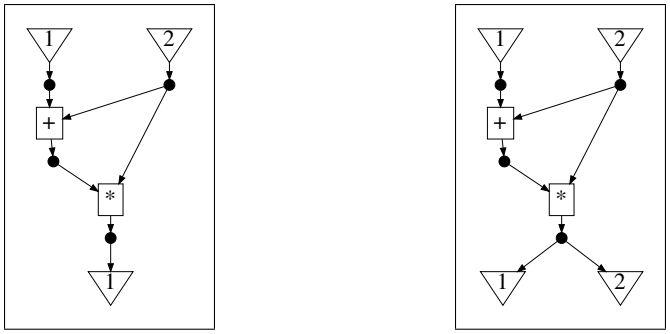
In Sect. 2, we present the adaptations we use to obtain a DPO-based definition of term graph transformation, and in Sect. 4 we sketch the proof that such transformation steps are semantics-preserving if the rule sides are semantically equivalent, after presenting some background on gs-monoidal categories of term graphs in Sect. 3.

## 2 Adapted DPO for Term Graph Rewriting

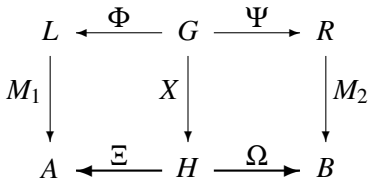
We are using the "jungle" view of term graphs, which goes back to Hoffmann and Plump (1991) and Corradini and Rossi (1993), since this is the view used by the gs-monoidal semantics, where nodes translate into objects and edges into morphisms.

When drawing such jungle term graphs, we start with the inputs on top and proceed down to the outputs, drawing nodes as bullets, and (hyper-)edges as labelled boxes connected to nodes via (implicitly ordered) input-tentacles and exactly one output-tentacle. (Although edges with multiple outputs have uses for example in the code graphs of (Kahl et al., 2006; Anand and Kahl, 2009), most of the literature, including all the cited work by Corradini and Gadducci, only considers single-output operations (edges), so we also do this here.) Graph input nodes are declared by attaching a triangle pointing to the input node — input nodes are necessarily distinct, and cannot be output nodes of edges. Graph input nodes are frequently called "variable nodes", and translated into distinct variables for a term reading. Graphs with multiple graph outputs are interpreted as standing for a tuple of terms; graph output nodes (in the literature frequently referred to as "roots") are declared by attaching a triangle pointing away from them — any node can be used as a graph output any number of times.

The left box in the following drawing depicts a term graph corresponding to the term "$(x_1 + x_2) * x_2$", while the term graph in the right box corresponds to "$((x_1 + x_2) * x_2, (x_1 + x_2) * x_2)$" or "let $z = (x_1 + x_2) * x_2$ in $(z, z)$":



We will use the following naming of graphs and morphisms in "DPO-shape" diagrams:




An example term graph transformation step in our adapted DPO approach is shown in Fig. 1 — in effect, this closely corresponds to the more low-level definitions of term graph transformation dominant in the literature: the "host graph" (or "context graph") $H$ can be thought of as obtained from the "application graph" $A$ by deleting all edges and inner nodes of $A$ which have a pre-image in $L$, but no pre-image (via $\Phi \fatsemi M_1$) in $G$, and the "result graph" $B$ is obtained from $H$ by "gluing in" the right-hand side $R$.

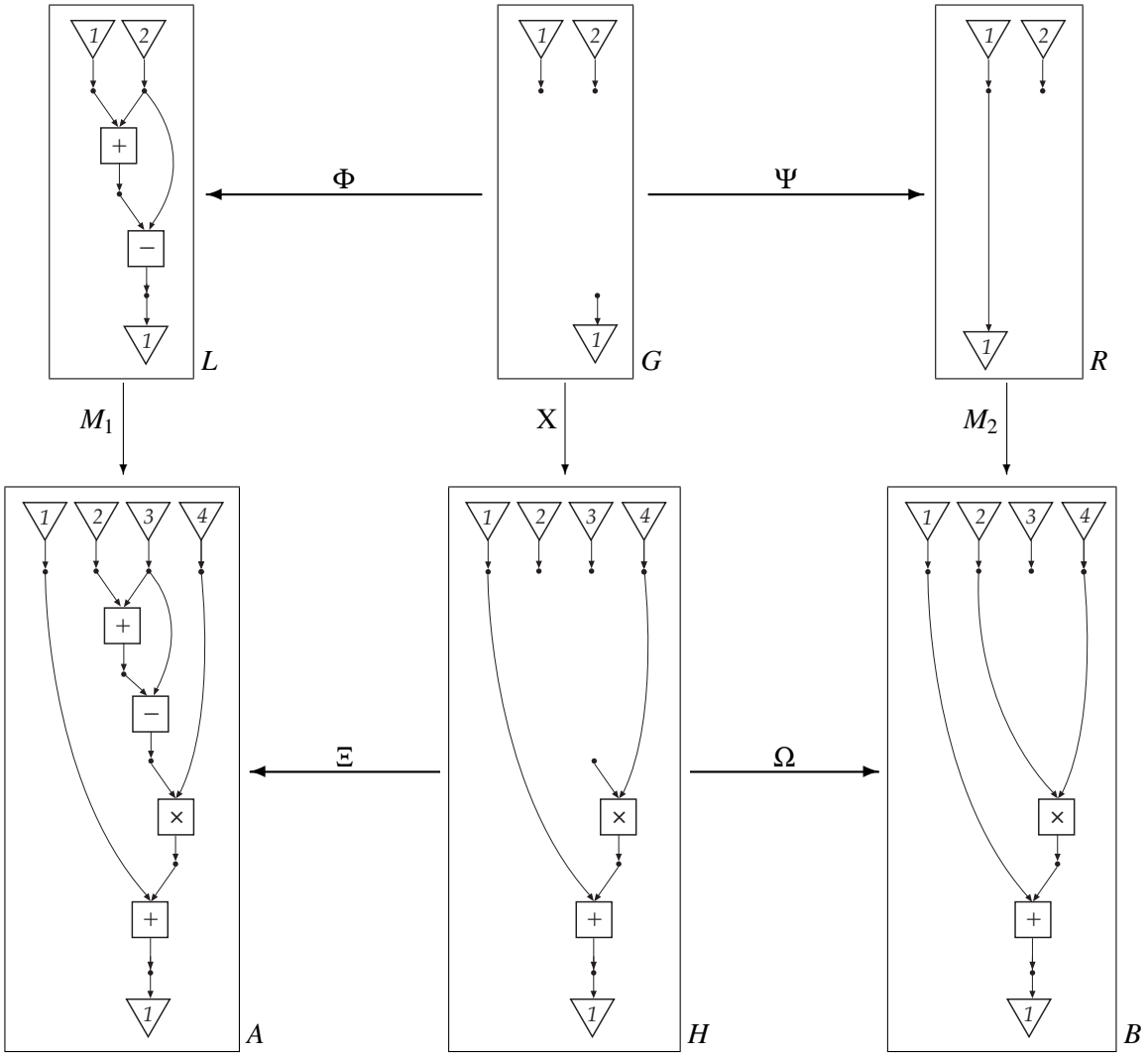



Figure 1: Example term graph rewriting step

The gluing graph $G$ and the host graph $H$ are obviously not jungles, since they have nodes that are neither graph input nodes nor edge output nodes, but they still are (ranked) *directed hypergraphs* (DHGs). In fact, we define jungles as *acyclic* DHGs where each non-input node is the output node of *exactly one* edge.

Both for DHGs and for jungles we distinguish *matchings*, which preserve edge labelling and incidence structure, from *homomorphisms*, which in addition preserve also graph input and output structure.

The diagram in Fig. 1 is then a double pushout in the category of DHG matchings, satisfying the following additional requirements:

- $M_1$ and $M_2$ are jungle matchings (which implies that $L$, $R$, $A$, and $B$ all are jungles),
- $\Phi$, $\Psi$, $\Xi$, $\Omega$ are DHG homomorphisms.

The existence of a pushout complement in the category of DHG matchings is subject to the gluing condition as usual — both dangling and identification conflicts can occur.

If the rule $L \xleftarrow{\Phi} G \xrightarrow{\Psi} R$ consists of DHG homomorphisms, both the pushout complement construction for the left square and the pushout construction for the right square will yield DHG matchings $\Xi$ and $\Omega$ that also respect the graph interface, and therefore are DHG homomorphisms.

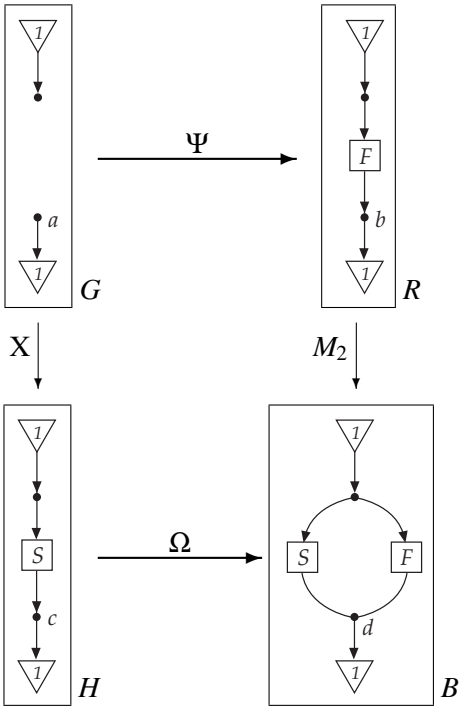



Figure 2: RHS edge conflict

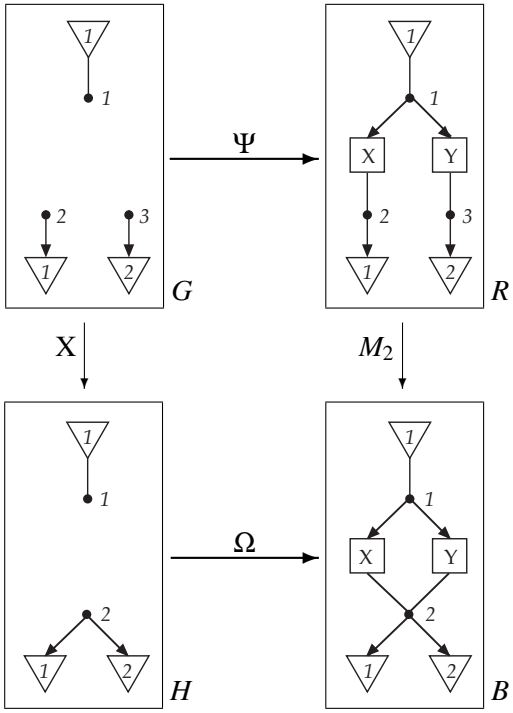



Figure 3: Non-injective host matching

For the right square of the DPO diagram, we finally have to ensure that $B$ is a jungle, which is not trivial. First, the situation shown in Fig. 2 would lead to $B$ not being a jungle — however, since the $\Phi$-image of node $a$ in $L$ has to be either an input node or the output of an edge, such a situation cannot occur at least when the rule LHS $\Phi$ is injective. (If the image of $a$ is an input node, then, with $\Phi$ preserving the graph interface, it cannot be injective. If the image of $a$ is the output node of an edge in $L$, then the image in $A$ of that edge needs to be also the image of the $S$-edge in $H$, which contradicts the left-hand pushout.) Second, also the example DHG matching pushout in Fig. 3 fails to produce a jungle $B$ — this situation

can be avoided by restricting the matching $M_1$ to be injective. (In effect, both constraints together correspond to the restriction to the "regular monos" of Corradini and Gadducci (2005, Prop. 4.3).)

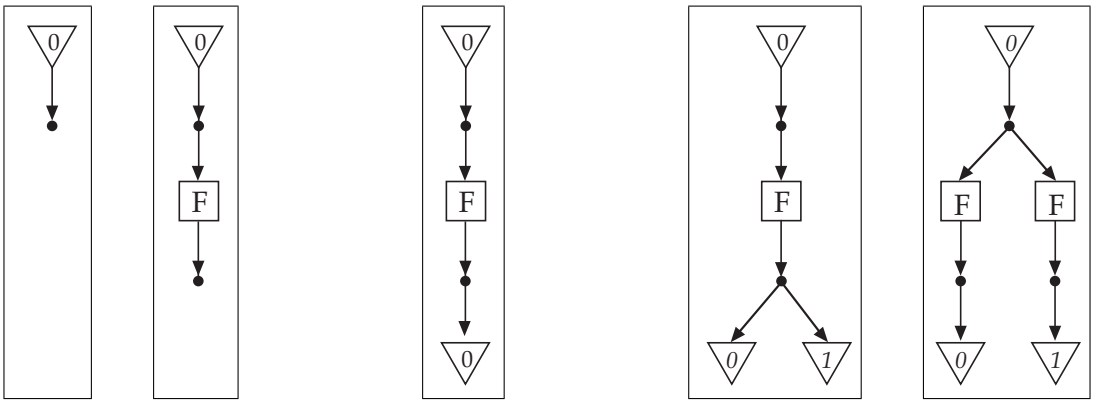
Since the right-hand side $\Psi$ of the rule is a DHG homomorphism, it is automatically injective on input nodes; non-injectivity of $\Psi$ therefore can only force identifications that are also "permissible" for the host graph, so we do not need to restrict $\Psi$ to be injective, which would be highly unwelcome for term graph rewriting.

Therefore, DPOs in the DHG matching category can be used to rewrite jungles with rules with injective left-hand sides, using only injective matchings.

## 3 Background on GS-Monoidal Categories of Term Graphs

Corradini and Gadducci (1999a) showed that term graphs (i.e., jungles) with sequential composition ($\fatsemi$) and parallel composition ($\otimes$) form a gs-monoidal category, where natural numbers (numbers of nodes in the graph input interface, respectively graph output interface) are objects, and term graphs with $m$ inputs and $n$ outputs are morphisms from $m$ to $n$.

The definition of gs-monoidal categories places them between symmetric monoidal categories and cartesian (monoidal) categories; the only difference with the latter is that, the "duplicator" transformation $\nabla$ producing diagonal maps $\nabla_A : A \to A \otimes A$ and the "terminator" transformation $!$ with components $!_A : A \to \mathbb{1}$ are both *not* assumed to be natural transformations (that is, for a morphism $F : A \to B$, the equations $F \fatsemi \nabla_B = \nabla_A \fatsemi (F \otimes F)$ and $F \fatsemi !_B = !_A$ do *not* necessarily hold.). For term graphs, the lack of naturality of $!$ means that *garbage* (nodes from which no output is reachable) makes a difference, such as between the two graphs to the left below, and the lack of naturality of $\nabla$ means that *sharing* (use of nodes in more than one consumer rôle, that is, as inputs for edges or as graph outputs) makes a difference, such as between the two graphs to the right below. (The words "garbage" and "sharing" motivate the name "gs-monoidal".)




$!_A$ $F \fatsemi !_B$ $F$ $F \fatsemi \nabla_B$ $\nabla_A \fatsemi (F \otimes F)$

Corradini and Gadducci (1999a) showed furthermore that the term graphs (jungles) over a given signature are arrows of the gs-monoidal category freely generated by that signature; therefore, there always exists a unique functor from the gs-monoidal category of term graphs to any gs-monoidal category. This induces a functorial semantics for term graphs (jungles) in any gs-monoidal category. (This will frequently be some (cartesian) category of sets, with some set $\mathcal{V}$ chosen as set of *values* "at a node"; a term graph with $m$ inputs and $n$ outputs then has a function of type $\mathcal{V}^m \to \mathcal{V}^n$ as semantics. For code genera-

tion applications, one may construct non-cartesian gs-monoidal semantics categories where morphisms contain information about resource usage, such as number of instructions.)

## 4 Semantics Preservation of DPO-Transformation of Term Graphs

While the fact that jungles form a free gs-monoidal category gives us semantics of jungles, it does not give us semantics of DHGs such as the gluing and host graphs in most typical rewriting steps. Rather than trying to artificially obtain some semantics for DHGs "with holes", we will transfer the necessary information "across the host graph $H$" at the DHG level.

A starting point could be the decomposition of term graphs into gs-monoidal expressions as described by Corradini and Gadducci (1999a). However, instead of extending this expression type into a type of contexts by including "placeholders" as proposed by Corradini and Gadducci (2002), we define contexts at the level of graphs:

**Definition 4.1** *An* $m,n$-context $(k,A_1,A_2)$ for an $i,j$-parameter *consists of:*

- *an* internal interface *object* $k$,
- *a* top part *jungle* $A_1 : m \to (i+k)$, *and*
- *a* bottom part *jungle* $A_2 : (j \otimes k) \to n$. □

In the following, we continue to use "$\fatsemi$" as sequential composition operator for term graphs, and "$\otimes$" for parallel composition. Furthermore, "$\mathbb{I}_k$" denotes the *identity* term graph with $k$ inputs that are also its outputs, in the same sequence. The empty DHG with $i$ inputs, and with $j$ distinct output nodes that are disjoint from the input nodes is written "$\bot_{i,j}$"; for the sub-category DHG *homomorphisms* restricted to DHGs with $i$ inputs and $j$ outputs, $\bot_{i,j}$ is the initial object.

By ensuring that there is no "side entrance" from within the application graph $A$ into the image of the LHS $L$, the dangling condition is crucial for the following result:

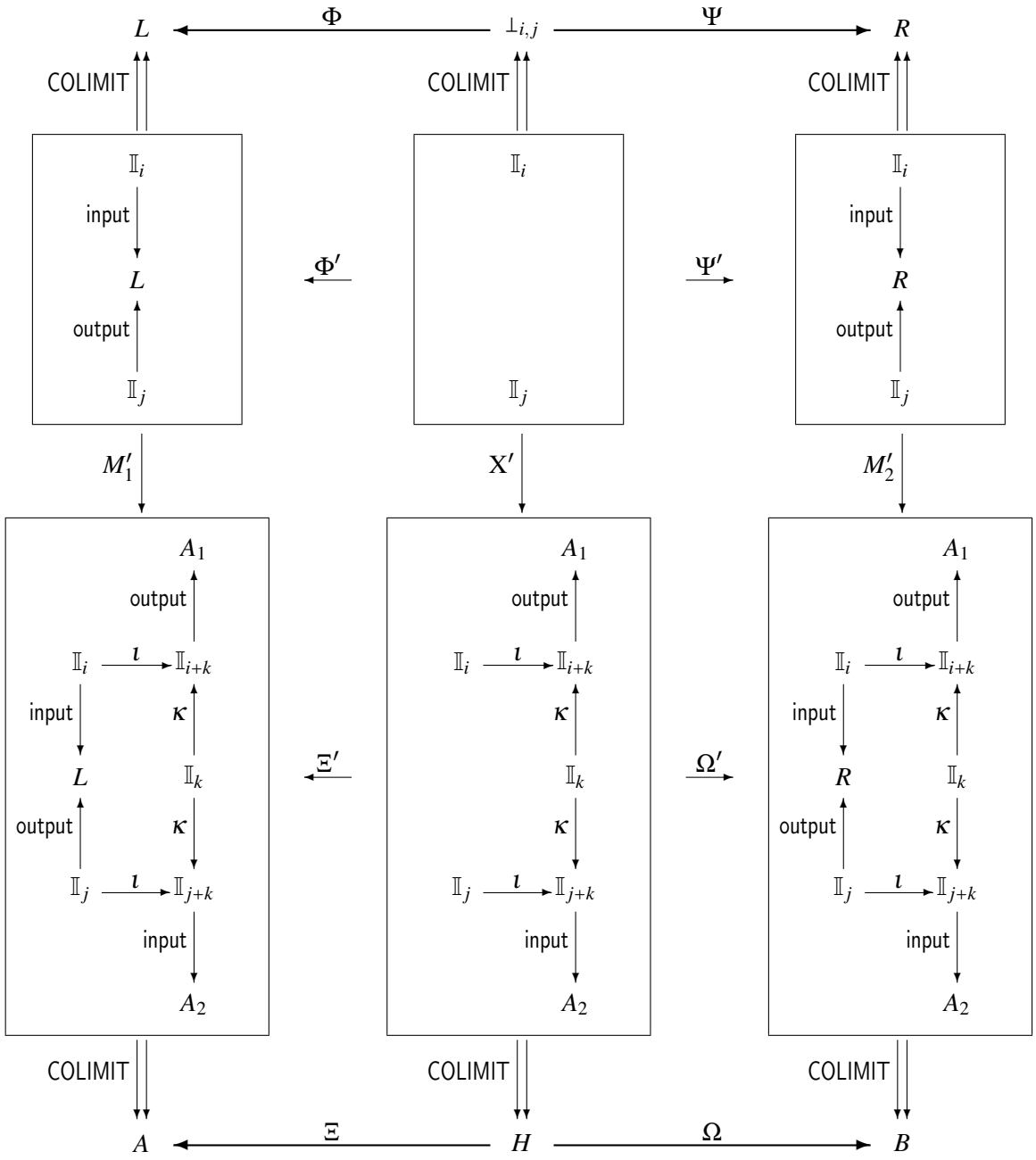
**Lemma 4.2** *Assume a jungle* $L : i \to j$ *be given, and let* $\Phi : \bot_{i,j} \to L$ *be the (necessarily-injective) DHG homomorphism from* $\bot_{i,j}$ *to* $L$. *If* $A : m \to n$ *is a jungle and* $M_1 : L \to A$ *is an injective jungle matching that together with* $\Phi$ *satisfies the dangling condition, then there is a* $m,n$*-context* $(k,A_1,A_2)$ *for an* $i,j$*-parameter such that* $A = A_1 \fatsemi (L \otimes \mathbb{I}_k) \fatsemi A_2$. □

Such a context can be calculated in several different ways from the reachability in $A$, for example by collecting all edges into $A_1$ that are reachable from the input nodes of $A$ via paths that do not touch the image of $L$ under $M_1$. The difference $A - A_1 - L$ would then induce $A_2$.

Sequential and parallel composition in the gs-monoidal category of term graphs (as morphisms) can be obtained as colimits in the category of DHG matchings. In the following diagram we denote the coproduct injections as $\iota$ and $\kappa$; for a DHG $X : m \to n$ we use $\mathsf{input} : \mathbb{I}_m \to X$ as the DHG matching mapping $\mathbb{I}_m$ identically to the input nodes of $X$, and analogously $\mathsf{output} : \mathbb{I}_m \to X$.

The lower left box below contains the diagram that has as its colimit the application graph $A$, factored into the context $(k,A_1,A_2)$ and an image of the left-hand side as $A = A_1 \fatsemi (L \otimes \mathbb{I}_k) \fatsemi A_2$.

The key observation is now that for a redex with $\bot$ as gluing graph and injective rule LHS $\Phi$ and injective matching $M_1$ satisfying the gluing condition, the DPO derivation step in the category of DHG matchings can be factored over a completely standard DPO diagram in the category of diagrams over the category of DHG matchings, as indicated in the following nested diagram:




A key ingredient for this to work is the restriction of the gluing graph to a "pure interface" $\bot_{i,j}$, so that it does not need to occur "in the place of $L$". It is crucial that this place is empty in the gluing and host diagrams, since otherwise we would not have proper diagram homomorphisms horizontally.

As a result, the context decomposition carries over to the result $B$ of the original DPO rewrite step, and we have:

$$B = A_1 \fatsemi (R \otimes \mathbb{I}_k) \fatsemi A_2$$

Let us now assume a semantics to be chosen, that is, some gs-monoidal category (e.g., *Set*), one of its objects $\mathcal{V}$ as interpretation of 1, and an appropriate monoidal bifunctor "_ × _". For a jungle $J : m \to n$, we write $[\![J]\!]_{m,n}$ for its semantics, which is a morphism from $\mathcal{V}^m$ to $\mathcal{V}^n$. In other words, we denote the

morphism component of the semantics functor with $[\![\_]\!]$; since this is a gs-monoidal functor, we have in particular $[\![J_1 \fatsemi J_2]\!] = [\![J_1]\!] \fatsemi [\![J_2]\!]$ and $[\![J_1 \otimes J_2]\!] = [\![J_1]\!] \times [\![J_2]\!]$, assuming ";" to stand for sequential composition in the semantics category.

Under the assumption that the rule $L \longleftarrow G \longrightarrow R$ is semantics preserving, that is, $[\![L]\!]_{i,j} = [\![R]\!]_{i.j}$, we therefore easily obtain semantics preservation of the rewrite result:

$$\begin{aligned}
[\![A]\!]_{m,n} &= [\![A_1 \fatsemi (L \otimes \mathbb{I}_k) \fatsemi A_2]\!]_{m,n} \\
&= [\![A_1]\!]_{m,i+k} \fatsemi ([\![L]\!]_{i,j} \times [\![\mathbb{I}_k]\!]_{k,k}) \fatsemi [\![A_2]\!]_{j+k,n} \\
&= [\![A_1]\!]_{m,i+k} \fatsemi ([\![R]\!]_{i,j} \times [\![\mathbb{I}_k]\!]_{k,k}) \fatsemi [\![A_2]\!]_{j+k,n} \\
&= [\![A_1 \fatsemi (R \otimes \mathbb{I}_k) \fatsemi A_2]\!]_{m,n} \\
&= [\![B]\!]_{m,n}
\end{aligned}$$

## 5 Conclusion and Outlook

By considering a straight-forward adaptation of the DPO approach to term graph rewriting, we obtained an easily-understandable concept of rule application. By lifting this adapted DPO into a standard DPO of diagrams, we have been able to transfer the context decomposition from the left-hand side to the right-hand side, obviating the need to consider any semantics for general DHGs such as $\bot_{i,j}$. As result, we obtain a semantics preservation theorem that will be an important tool in the generation verified code optimisation tools employing rule-based transformation of data-flow graphs, as outlined for example in (Kahl, 2014).

## References

C. K. Anand, W. Kahl. An optimized Cell BE special function library generated by Coconut. *IEEE Transactions on Computers*, 58(8):1126–1138, 2009. doi: 10.1109/TC.2008.223.

Z. M. Ariola, J. W. Klop, D. Plump. Bisimilarity in term graph rewriting. *Information and Computation*, 156(1):2–24, 2000. ISSN 0890-5401. doi: 10.1006/inco.1999.2824.

R. Banach. A fibration semantics for extended term graph rewriting. In Sleep et al. (1993), chapt. 7, pp. 91–100.

A. Corradini, F. Gadducci. A 2-categorical presentation of term graph rewriting. In E. Moggi, G. Rosolini (eds.), *Category Theory and Computer Science*, pp. 87–105, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. doi: 10.1007/BFb0026983.

A. Corradini, F. Gadducci. An algebraic presentation of term graphs, via gs-monoidal categories. *Applied Categorical Structures*, 7(4):299–331, 1999a. ISSN 1572-9095. doi: 10.1023/A:1008647417502.

A. Corradini, F. Gadducci. Rewriting on cyclic structures: Equivalence between the operational and the categorical description. *RAIRO Theor. Inform. Appl.*, 33:467–493, 1999b. doi: https://doi.org/10.1051/ita:1999128.

A. Corradini, F. Gadducci. Categorical rewriting of term-like structures. *ENTCS*, 51:108–121, 2002. ISSN 1571-0661. doi: 10.1016/S1571-0661(04)80195-6. GETGRATS Closing Workshop.

A. Corradini, F. Gadducci. On term graphs as an adhesive category. *ENTCS*, 127(5):43–56, 2005. doi: https://doi.org/10.1016/j.entcs.2005.02.014.

A. Corradini, F. Rossi. Hyperedge replacement jungle rewriting for term-rewriting systems and logic programming. *Theoret. Comput. Sci.*, 109(1–2):7–48, 1993. doi: 10.1016/0304-3975(93)90063-Y.

D. Duval, R. Echahed, F. Prost. A heterogeneous pushout approach to term-graph transformation. In R. Treinen (ed.) RTA 2009, pp. 194–208, Berlin, Heidelberg, 2009. Springer.

A. Habel, D. Plump. $\mathcal{M},\mathcal{N}$-adhesive transformation systems. In H. Ehrig et al. (eds.) ICGT 2012, LNCS, vol. 7562, pp. 218–233. Springer, 2012. doi: 10.1007/978-3-642-33654-6_15.

B. Hoffmann, D. Plump. Implementing term rewriting by jungle evaluation. *Informatique théorique et applications/Theoretical Informatics and Applications*, 25(5):445–472, 1991.

W. Kahl. *Algebraische Termgraphersetzung mit gebundenen Variablen*. Reihe Informatik. Herbert Utz Verlag Wissenschaft, München, 1996. ISBN 3-931327-60-4; also Doctoral Diss. at Univ. der Bundeswehr München, Fakultät für Informatik.

W. Kahl. A fibred approach to rewriting — how the duality between adding and deleting cooperates with the difference between matching and rewriting. Technical Report 9702, Fakultät für Informatik, Universität der Bundeswehr München, 1997. URL `http://www.cas.mcmaster.ca/~kahl/Publications/TR/Kahl-1997b.html`.

W. Kahl. Towards "mouldable code" via nested code graph transformation. *J. Logic and Algebraic Programming*, 83(2):225–234, 2014. doi: 10.1016/j.jlap.2014.02.010.

W. Kahl, C. K. Anand, J. Carette. Control-flow semantics for assembly-level data-flow graphs. In W. McCaull, M. Winter, I. Düntsch (eds.), *8th Intl. Seminar on Relational Methods in Computer Science, RelMiCS 8, Feb. 2005*, LNCS, vol. 3929, pp. 147–160. Springer, 2006. doi: 10.1007/11734673_12.

J. R. Kennaway, J. W. Klop, M. R. Sleep, F. J. de Vries. The adequacy of term graph rewriting for simulating term rewriting. In Sleep et al. (1993), chapt. 12, pp. 157–170.

J. R. Kennaway, J. W. Klop, M. R. Sleep, F. J. de Vries. On the adequacy of graph rewriting for simulating term rewriting. *ACM Trans. Prog. Lang. Syst.*, 16(3):493–523, 1994.

F. W. Lawvere. Functorial semantics of algebraic theories. *Proc. Nat. Acad. Sci. USA*, 50:869–872, 1963.

D. Plump. Essentials of term graph rewriting. *ENTCS*, 51:277–289, 2002. ISSN 1571-0661. doi: 10.1016/S1571-0661(04)80210-X. GETGRATS Closing Workshop.

K. H. Rose. Graph-based operational semantics of a lazy functional language. In Sleep et al. (1993), chapt. 22, pp. 303–316.

M. R. Sleep, M. J. Plasmeijer, M. C. J. D. van Eekelen (eds.). *Term Graph Rewriting: Theory and Practice*. Wiley, 1993.

C. P. Wadsworth. *Semantics and Pragmatics of the Lambda Calculus*. D.Phil. thesis, Oxford University, 1971.