

---

# From Trees To Graphs: Understanding The Implications Of Sharing For Rewriting

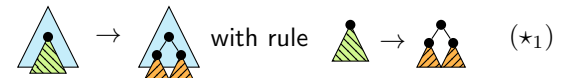
**Student. Maria A Schett**  
University of Innsbruck  
Technikerstrasse 21a  
6020 Innsbruck, Austria  
maria.schett@student.uibk.ac.at

ACM Student Membership  
Number: 7746806  
Category: Master Student

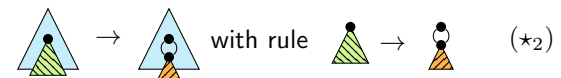
**Advisor. Georg C Moser**  
University of Innsbruck  
Technikerstrasse 21a  
6020 Innsbruck, Austria  
georg.moser@uibk.ac.at

## Problem & Motivation

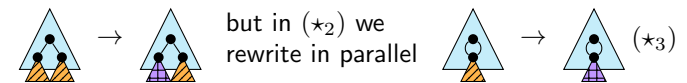
Rewriting is the transformation of objects based on a set of directed equations, namely rewrite rules. If these objects are terms, we talk about term rewriting—a Turing-complete, yet simple, abstract model of computation [2, 9]. We rewrite:



Inherent to terms is their tree structure and thereby rewrite steps can cause an exponential blow-up in size. To avoid this we move from trees to graphs and share equal sub-terms:



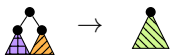
Now with the rule  $\text{orange} \rightarrow \text{purple}$  in  $(\star_1)$  we can choose to rewrite only the first argument:



We see that sharing influences the possible rewrite steps. My research aims at understanding this influence. In particular I want to understand the influence of sharing on termination, i.e., the absence of infinite rewrite sequences.

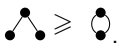
## Background & Related Work

Term graph rewriting with explicit sharing *and* the reverse operation—unsharing—can simulate term rewriting with polynomial overhead and linear size growth [4, 1]. But simulation requires unsharing which seems to annihilate the advantage of the efficient graph representation. Thus [8] investigates the rewriting relation with only an explicit sharing operation. Unsurprisingly, every term graph rewrite step can be simulated by  $n$  term rewrite steps ( $n \geq 1$ ), but not vice versa ( $\star_3$ ). As a direct consequence we have that termination of term rewriting implies termination of term graph rewriting.

E.g., if we add the rule:  to  $(\star_1)$  and  $(\star_2)$ , we have termination for  $(\star_2)$  but not for  $(\star_1)$ . I am interested in this gap and want to find techniques to automatically prove termination of term graph rewriting (in the absence of termination of term rewriting). The only technique, to my knowledge, is developed by [7]. In my work I follow that idea. I extend it by transferring the results to the term graph rewriting formalism of [1] and re-proving the result.

## Approach & Uniqueness

Many methods that prove termination rely on finding a well-founded order such that every rewrite step corresponds to a decrease in this order, e.g., Knuth-Bendix order or some recursive path orders. Proofs that these orders are well-founded are usually based on Kruskal's Tree Theorem [5]: if we can find an order on the symbols in a tree, we can extend this order to an order on trees. Following [7] our symbols are *tops*. The top of a term graph is its root *and* its direct successors—thus keeping information on how these successors are shared.

E.g., in  $(\star_2)$  we could define the order on tops: . This sharing of successors gives us an additional way of

distinguishing symbols. Crucial here is the absence of unsharing. In [7] relies on an encoding of tops to symbols and uses Kruskal's tree theorem, but hints that there is a direct proof based on [6]. This is the starting point for my work.

## Results & Contributions

A first step is to prove Kruskal's tree theorem directly for term graphs to understand the effects of sharing for termination, and as future work also complexity bounds on the amount of possible rewrite steps. This I achieve by following the a minimal bad sequence argument of Nash-Williams [6]: assuming the existence of a minimal "bad" infinite sequence, an even smaller "bad" infinite sequence is constructed contradicting minimality. The most important insight from my proof concerns the arguments of a term graph—or rather *the* argument. For a term structure we have several sub-terms, i.e., sub-trees, as arguments. For a term graph structure it is beneficial to regard the arguments as only a single argument graph. This preserves sharing. Moreover a single argument simplifies the proof as extending the order to sequences, Higman's Lemma [3], can be omitted.

Graphs are ubiquitous in computer science. We find them in flow graphs, heaps, modeling languages. . . Thus I believe that working on the foundations of term graphs as well as an automatic analysis of transformations, i.e., rewriting, is vital.

## References

- [1] Avanzini, M. *Verifying Polytime Computability Automatically*. PhD thesis, University of Innsbruck, 2013.
- [2] Baader, F., and Nipkow, T. *Term Rewriting and All That*. Cambridge Univ. Press, 1998.
- [3] Higman, G. Ordering by Divisibility in Abstract Algebras. *Proc. of the London Mathematical Society* 3, 2 (1952), 326–336.
- [4] Kennaway, J. R., Klop, J. W., Sleep, M. R., and de Vries, F. J. On The Adequacy Of Graph Rewriting For Simulating Term Rewriting. *ACM Trans. Program. Lang. Syst.* 16, 3 (1994), 493–523.
- [5] Kruskal, J. B. Well-Quasi-Ordering, The Tree Theorem, and Vazsonyi's Conjecture. *Transactions of the AMS* 95, 2 (1960), 210–225.
- [6] Nash-Williams, C. S. J. A. On Well-Quasi-Ordering Finite Trees. *Math. Proc. Cambridge Philosophical Society* 59 (1963), 833–835.
- [7] Plump, D. Simplification Orders for Term Graph Rewriting. In *Mathematical Foundations of Computer Science* (1997), 458–467.
- [8] Plump, D. *Handbook of Graph Grammars and Computing by Graph Transformation*, vol. 2. World Scientific, 1999, ch. Term Graph Rewriting, 3–61.
- [9] TeReSe, Ed. vol. 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.