

Direct Combination of Completion and Congruence Closure

Leo Bachmair¹ and Christelle Scharff^{2*}

¹ State University of New York at Stony Brook, Department of Computer Science,
Stony Brook, NY, USA, leo@cs.sunysb.edu

² Pace University, Department of Computer Science, New York, NY, USA,
cscharff@pace.edu

Theories presented by finite sets of ground (i.e., variable-free) equations are known to be decidable. There are different approaches to dealing with ground equational theories. Researchers interested in term rewriting, for instance, have applied *completion*, a method that transforms a given set of equations into a set of directed rules that is both terminating and confluent; see [3] for an overview. Such a convergent rewrite system defines unique normal forms for equal terms and hence provides a decision procedure for the word problem of the underlying equational theory. Completion itself is a semi-decision procedure but under certain reasonable assumptions about the strategy used to transform equations, is guaranteed to terminate if the input is a set of ground equations.

A rather distinct approach to solving word problems for ground equational theories is based on the computation of the so-called *congruence closure* of a relation. Various efficient congruence closure algorithms have been proposed. They typically use a compact representation of the given terms by a directed acyclic graph; cf., [6, 4, 8, 10]. Completion methods are usually not as efficient, though one efficient ground completion method has been described by [11], who obtains an $O(n \log n)$ algorithm that cleverly uses congruence closure to transform a given set of ground equations into a convergent ground rewrite system. The standard completion approach is quadratic in the worst case [9].

In this work we bridge the gap between the two approaches and provide a better understanding of the connection between completion and congruence closure by defining a new, efficient congruence closure method that integrates key ideas of completion and graph-based methods in a novel, direct and natural way. Our method takes its origin in the abstract congruence closure of [1] and the SOUR-graph method of [7]. It may be viewed as a modified version of the SOUR-graph approach, but with simpler graphs. In contrast to SOUR graphs it is not always possible to

* Supported by a Pace University CSIS research summer grant.

extract a (convergent) rewrite system from these graphs, but one always obtains an (partially constructed) abstract congruence closure, that is, a convergent rewrite system over an *extended* signature.

In a SOUR graphs the vertices represent terms and the edges carry information about subterm relations between terms (S), rewrite rules (R), unifiability of terms (U) and order relations between terms (O). We consider only ground terms and hence do not need unification edges. We also dispense with order edges and manipulate rewrite edges in a different way, based on the explicit use of edges (E) representing unordered equations. (In short, our modifications amount to what might be called “SER graphs.”) The deletion of the order edges is crucial for reasons of efficiency: It is a characteristic advantage of our method—and, of course, of (abstract) congruence closure algorithms in general—that no explicit term ordering needs to be computed. (The corresponding disadvantage is that we do not obtain a convergent rewrite system on the original signature, but only over an extended signature.) The graph transformation rules we use are simpler versions of SOUR graph rules and essentially represent (combinations of) critical pair computations and term simplifications. We use fewer rules and also explicitly formulate a “merge rule” that is well-known from congruence closure algorithms, but only implicitly used in SOUR graphs.

Specifically, we represent terms and equations by a directed acyclic graph (DAG), where each vertex is labeled by a symbol from the initial term signature Σ and edges are classified as E (equation), R (rewrite), or S (subterm). In addition, each vertex has a name, which we view as a constant; the set K of all such constants is required to be disjoint from Σ . This shared representation is inherently more efficient than any other representation of terms. Each vertex represents a term over the extended signature $\Sigma \cup K$. It also represents a term over the initial signature Σ , but the representation schema is not as direct as in the case of SOUR graphs and, due to the absence of a term ordering, it may make no sense to interpret rewrite rules over the original signature. The graph transformation rules implement a version of abstract congruence closure. We may (i) *orient* an equation, (ii) compute critical pairs (RR rule), (iii) simplify terms (SR rule), or (iv) *merge* nodes to ensure closure under congruence. Exhaustive application of these transformation rules terminates, is sound in that the equational theory represented over Σ -terms does not change, and complete in the sense that the final rewrite system over the extended signature is convergent.

This “congruence closure phase” does not produce a convergent rewrite system over the original signature, but we may obtain such a system by further transforming the graph in a way reminiscent of the compression and selection rules of [2]. These rules eliminate vertices from the graph (and constants from K) and redirect R edges. Their exhaustive application produces a graph representing a convergent rewrite system over the initial signature.

There is a link between unification and congruence closure [5]; unification closure can be reduced to a special form of congruence closure. We plan to explore this connection between unification and congruence closure in the graph-based framework of our method.

A long version of the paper is available at:

<http://www.csis.pace.edu/~scharff>.

References

- [1] Bachmair, L. and Ramakrishnan, I.V. and Tiwari, A. and Vigneron, L. Congruence Closure Modulo Associativity and Commutativity. In Kirchner, H. and Ringeissen, C. editors, *Frontiers of Combining Systems, 3rd Intl Workshop Fro CoS 2000*, Volume 1794 of *Lecture Notes in Artificial Intelligence*, pages 245-259, Nancy, France, March 2000, Springer-Verlag.
- [2] Bachmair, L. and Tiwari, A. and Vigneron, L. Abstract Congruence Closure. *To appear in Journal of Automated Reasoning*, 2002.
- [3] Dershowitz, N. and Jouannaud, J.-P. *Handbook of Theoretical Computer Science*, volume B, chapter 6: Rewrite Systems, pages 244-320, Elsevier Science Publishers B.V. (North Holland), 1990, also as Research report 478, LRI.
- [4] Downey, P. J. and Sethi, R. and Tarjan, R.E. Variations on the common subexpressions problems. *JACM*, 27(4), pages 758-771, 1980.
- [5] Kanellakis, P. and Revesz, P. On the Relationship of Congruence Closure and Unification. *Journal of Symbolic Computation*, 7(3& 4):427-444, 1989, Special issue on unification, part one.
- [6] Kozen, D. Complexity of finitely presented algebras. In *Proceedings of 9th Symposium of Theory of Computation*, pages 164-177, May, 1977.
- [7] Lynch, C. and Strogova, P. SOUR Graphs for Efficient Completion. *Discrete Mathematics and Theoretical Computer Science*, volume 2, pages 1-25, 1998.
- [8] Nelson, G. and Oppen, D.C. Fast Decision Procedures Based on Congruence Closure. *JACM*, 27(2):356-364, 1980.
- [9] Plaisted, D. A. and Sattler-Klein, A. Proof Lengths for Equational Completion. *Information and Computation*, 125(2):154-170, 1996.
- [10] Shostak, R.E. Deciding Combinations of Theories. *JACM*, 31(1):1-12, 1984.
- [11] Snyder, W. A Fast Algorithm for Generating Reduced Ground Rewriting Systems from a Set of Ground Equations. *Journal of Symbolic Computation*, 15(4):415-450, 1993.