Detecting Modular ACU Structural Symmetries*

M. Alpuente[1], S. Escobar[1] and J. Espert[1]

DSIC-ELP, Universitat Politècnica de València, Spain
{alpuente,sescobar,jespert}@dsic.upv.es

Abstract

We present an efficient encoding of order-sorted modular ACU terms into colored directed graphs. Then, by computing the automorphism groups of the encoded graphs, we are able to extract modular ACU structural symmetries both inside a term and across a set of terms. Finally, we show how the computed symmetries can be applied to the optimization of the equational generalization algorithms for modular ACU theories.

1 Introduction

Generalization (also known as anti-unification) is the dual of unification [7]. Roughly speaking, the generalization problem for two terms t_1 and t_2 means finding their *least general generalization* (lgg), i.e., the least general term t such that both t_1 and t_2 are instances of t under appropriate substitutions. While studying order-sorted modular ACU[1] generalization in [1], we found that our algorithm spends a large amount of time performing computations that, although different from previous ones, are somehow symmetrical to them and lead to equivalent (and thus redundant) results. Since this problem cannot be mitigated with mere memoization, we set out to find a way of preprocessing the terms to detect symmetries and instrument our algorithm to exploit them. This paper presents that ongoing work; in particular, we present a technique to detect modular ACU symmetries among a set of terms. We formulate an injective encoding of order-sorted modular ACU terms in flat normal form into graphs with colored nodes. Then, we extract useful term redundancies from the automorphism groups of these graphs, which can themselves be computed using efficient, well-known algorithms [6, 8]. By imposing an arbitrary order among equivalent subterms based on the automorphism groups, we avoid generating many redundant computations in the equational least general generalization algorithm that we formulated in [1].

Let us illustrate the problem with some equational generalization examples. Let f be an associative and commutative function symbol over natural numbers[2]. Consider the least general generalization problem of the terms $f(0, 1, 2, 3, 4)$ and $f(0, 5, 6, 7, 8)$. This problem has a single solution: $f(0, x, y, z, u)$, where x, y, z, and u are fresh variables of sort Nat because the constant 0 is the only term that is shared by both terms and the other constants have the same sort (Nat) and only appear in one of the two given terms.

[*] M. Alpuente, S. Escobar, and J. Espert have been partially supported by the EU (FEDER) and the Spanish MEC/MICINN under grant TIN 2010-21062-C02-02, and by Generalitat Valenciana PROMETEO2011/052. J. Espert has also been supported by the Spanish FPU grant FPU12/06223.

[1] Modular ACU generalization supports any combination of associative (A), commutative (C), and unity (U) equational attributes for the function symbols of a given signature. By ACU (not qualified by modular) we refer to theories with the three types of equational attributes. In the same way, AC generalization refers to theories that satisfy associativity and commutativity properties, but not unity.

[2] Ignore Peano or other notations for now, just think of each number as a different constant.

Let us consider now the least general generarization of $f(0,0,1,1,2,2,3,3,4,4)$ and $f(0,0,5,5,6,6,7,7,8,8)$, where the different constants occur twice each; in this case, their least general generalizer is the term $f(0,0,x,x,y,y,z,z,t,t)$. Observe that, like in the previous example, we do not care which non-zero number each variable of the generalizer refers to; i.e., the choice between non-zero constants in this example is indifferent w.r.t. term structure.

However the problem becomes more involved if we have different symbols with possibly different equational attributes as in the case of the least general generalization problem for the terms $f(0,1,2,g(0,1),g(1,2),g(2,0))$ and $f(3,4,5,g(3,4),g(4,5),g(5,3))$, where g is a commutative, but not associative symbol. The least general generalizer for this problem is $f(x,y,z,g(x,y),g(y,z),g(z,x))$. It is interesting to note how the possible generalizer substitutions reveal these symmetries: from the least general generalizer, we can obtain the first term $f(0,1,2,g(0,1),g(1,2),g(2,0))$ by applying any of the substitutions $\{\, x \to 0, y \to 1, z \to 2 \,\}$, $\{\, x \to 1, y \to 0, z \to 2 \,\}$, $\{\, x \to 2, y \to 1, z \to 0 \,\}$, *etc.* In other words, the only requirement is that the mapping from $\{\, x, y, z \,\}$ to $\{\, 0, 1, 2 \,\}$ introduced by the substitution be bijective; the mapping itself is irrelevant, since 0, 1, and 2 are *structurally equivalent.*

2 Graph automorphisms

Isomorphisms of graphs [5] are bijections of the vertex sets preserving adjacency as well as non-adjacency. In the case of directed graphs, orientations must be preserved; in the case of graphs with colored edges and/or vertices, colors must also be preserved. *Automorphisms* of the graph $X = (V, E)$ are $X \to X$ isomorphisms; they form the subgroup $Aut(X)$ of the symmetric group $Sym(V)$. Automorphisms of directed graphs are defined analogously.

The connection between graph isomorphisms and terms with equational attributes has been previously explored. In [3], Basin shows that AC term equivalence under variable renaming can be reduced to graph isomorphism and proves that both problems have the same complexity. In [2], Avenhaus and Plaisted formulate a technique based on graph isomorphisms to reduce the search space in equational inference problems. The study of algorithms for the computation of isomorphisms and automorphism groups and their computational implementation is a mature but active area of research. We rely on Bliss—a state-of-the-art automorphism group computation tool [6]—for our prototypical implementations.

3 The encoding

In this paper, we define an encoding of order-sorted modular ACU terms [4] in flat normal form (see [1]) into colored directed graphs that satisfy the following properties:

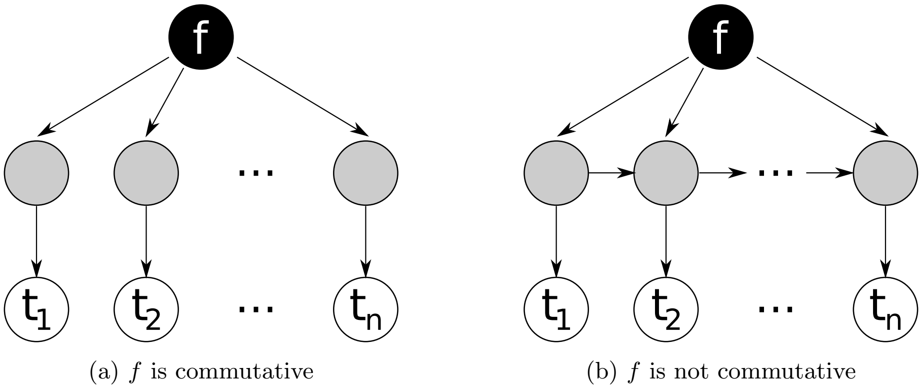
1. Every well-formed term can be encoded into a valid graph.

2. The resulting graphs preserve the equational properties of the original term.

3. That information that we extract from the automorphisms of the encoded graphs always correspond with valid structural symmetries of the original terms.

We assume the existence of a mapping color(x) from $\Sigma \cup \mathcal{X}$ to the set of the natural numbers. The elements of the image set of this mapping are the possible *colors* of the nodes. This mapping can be any one that obeys the following rules:

1. Variables with the same sort are mapped to the same number.



(a) f is commutative (b) f is not commutative

Figure 1: Encoding of $f(t_1, \dots, t_n)$

2. Constants with the same sort are mapped to the same number.

3. Function symbols with non-zero arity are mapped to a unique number that is not shared with any other symbol, regardless of its sort.

4. Two elements are not mapped to the same number, unless they verify one of the previous conditions.

Our encoding codomain has two classes of graph nodes: nodes that are *labelled* with a function symbol and unlabelled *auxiliary* nodes. Auxiliary nodes are always connected by an outgoing arc to a labelled node (child node) and by an ingoing arc to a different labelled node (parent node). Auxiliary nodes whose parent node is labelled with a non-commutative (free, A, U, or AU) symbol can have additional links among them; this allows us to enforce ordering constraints. All auxiliary nodes share the same, special color. We formulate our encoding as follows:

1. Each variable is encoded into a unique node that is shared by all occurrences of the variable.

2. Each constant is encoded into a unique node that is shared by all occurrences of the constant.

3. Let f be a commutative (C, AC, CU, or ACU) symbol and $f(t_1, \dots, t_n)$ be a term. The root symbol f is encoded as a node with outgoing arcs pointing to the root symbols of the encoded subterms $t_1, \dots, t_n$, using fresh auxiliary nodes as intermediaries. Figure 1a depicts the structure of this transformation.

4. Let f be a non-commutative (free, A, U, or AU) symbol and $f(t_1, \dots, t_n)$ be a term. The root symbol f is encoded as a node. Similarly to the commutative case, the root symbol f is encoded as a node with outgoing arcs pointing to the root symbols of the encoded subterms $t_1, \dots, t_n$, using fresh auxiliary nodes as intermediaries. The difference with the preceeding case is that each auxiliary node has an outgoing arc to the auxiliary node of the next subterm (except for the last term), which allows us to express ordering constraints. Figure 1b depicts the structure of this transformation.

Table 1: AC-decompositions with and without preprocessing

size (S)	8	10	12	14	16
unoptimized (U)	15	184	2945	63756	(error)
optimized (O)	5	29	209	1909	20801
ratio (U/O)	3	6.3	14.4	33	-

The reader might think that the auxiliar nodes are redundant in the commutative cases (C, AC, CU, and ACU). However, the resulting, simplified encoding would be incompatible with term sharing. Consider the term $f(0, 0, 1)$, where f is an AC function symbol and 0 are 1 constants of the sort Nat. The encoded graph would have two identical arcs from f to the node that represents the encoding of the constant 0, which would be an invalid graph. Note that the duplication of the node associated to the constant 0 would not lead to a valid representation either, because then the encoded form would not be able to distinguish among constants that belong to the same sort, as is the case for the constants 0 and 1.

The U axioms are ignored for encoding purposes. In flat normal form, the U axioms lead to compact representations (minimum length) that do not introduce structural redundancies. Hence, we can detect the symmetries for ACU function symbols as if they were AC, AU symbols as if they were A, CU symbols as if they were C, and U symbols as if they were free.

4 Applications

By extending our signature with a fresh commutative symbol $\sqcup$ with an appropriate sort signature, we can combine the encodings of a set of terms $\{ t_1, \ldots, t_n \}$ to deliver the encoding of the term $\sqcup(t_1, \ldots, t_n)$. This simple trick allows us to detect modular ACU symmetries across terms as well as internal symmetries that are consistent across all the given terms. These symmetries are relevant for the equational generalization of modular ACU theories. Table 1 shows how the symmetries can avoid a large number of computations in the computation of AC-decompositions, a critical part of our equational generalization algorithm. For this experiment we have used two terms of the form $f(0, \ldots, (n-1))$ and $f(n, ..., (2n-1))$ where f is an associative-commutative function symbol and the numbers represent different constants. This configuration appears frequently when the input terms, or part of them, represent sets. In Table 1, row (S) indicates the original size (number of symbols) of the terms, row (U) represents the number of AC-decompositions that must be checked by the original, unoptimized algorithm and row (O) stands for the number of AC-decompositions that must be checked after using our symmetry-based transformation. Note that the achieved speedup (U/O) grows exponentially with the size of the terms.

As an example, Figure 2 shows the encoding for the (unsorted) generalization problem Γ of terms $f(g(a, b), g(b, c))$ and $f(g(b, c), g(c, b))$, where a, b, and c are constants, f is an associative-commutative function, and g is an associative but not commutative function. The dashed lines link structurally equivalent nodes, as computed by the technique. Observe that this generalization problem is symmetric, because the top-level symbols of both terms are linked. We can transform one term into the other by swapping constants a and c and applying equational axioms. This property halves the number of cases to consider in the generalization algorithm.

Finally, we would like to remark that although this technique has been developed to optimize the computation of lggs, the technique itself is agnostic w.r.t. to generalization and we believe

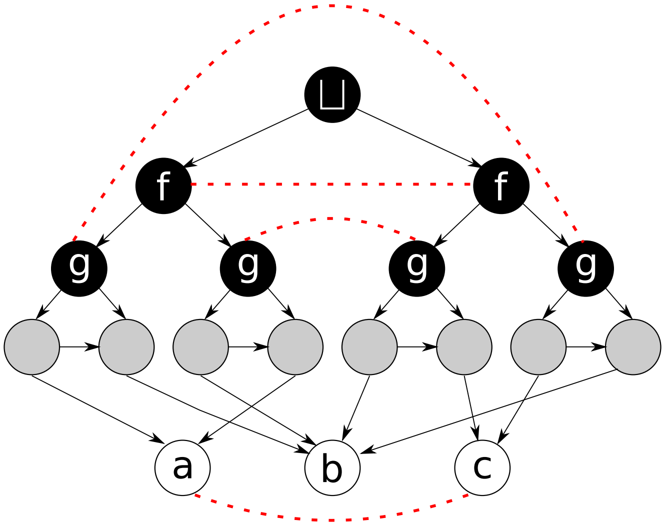


Figure 2: Encoding of the generalization problem Γ

that it might be useful in other domains—such as equational unification—that might benefit from discovering the structural symmetries introduced by equational attributes. As future work, we plan to study the relationship between the symmetries than can be detected and exploited in equational AC-unification by following our methodology compared to more conventional equational unification algorithms based on diophantine encoding.

Acknowledgements

We thank anonymous reviewers for their useful insights and for suggesting bibliography on related automated reasoning techniques.

References

[1] M. Alpuente, S. Escobar, J. Espert, and J. Meseguer. A modular order-sorted equational generalization algorithm. *Information and Computation*, To appear.

[2] Jürgen Avenhaus and David A. Plaisted. General algorithms for permutations in equational inference. *IFIP Trans. A.*, 26:2001, 2001.

[3] David A. Basin. A term equality problem equivalent to graph isomorphism. *Inf. Process. Lett.*, 51(2):61–66, 1994.

[4] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude - A High-Performance Logical Framework*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[5] R. L. Graham, M. Grötschel, and L. Lovász, editors. *Handbook of combinatorics (vol. 2)*. MIT Press, Cambridge, MA, USA, 1995.

[6] Tommi Junttila and Petteri Kaski. Engineering an efficient canonical labeling tool for large and sparse graphs. In *Proc. ALENEX07*, pages 135–149. SIAM, 2007.

[7] G. D. Plotkin. A note on inductive generalization. *Mach. Intell.*, 5:153–163, 1970.

[8] Jacobo Torán. On the hardness of graph isomorphism. *SIAM J. Comput.*, 33(5):1093–1108, May 2004.