A SAT+CAS Method for Enumerating Williamson Matrices of Even Order

Curtis Bright, University of Waterloo
Ilias Kotsireas, Wilfrid Laurier University
Vijay Ganesh, University of Waterloo

Brute-brute force has no hope. But clever, inspired brute force is the future. –Doron Zeilberger

Motivation

Many mathematical conjectures concern the existence or nonexistence of combinatorial objects that are only feasibly constructed through a search. To find large instances of these objects, it is necessary to use a computer with a clever search procedure.

Example: Williamson Conjecture

Symmetric sequences A, B, C, D of length n are called *Williamson* if they have ±1 entries and satisfy

$$\mathrm{PSD}_A(s) + \mathrm{PSD}_B(s) + \mathrm{PSD}_C(s) + \mathrm{PSD}_D(s) = 4n \quad (*)$$

for all integers s, where PSD_X is the squared absolute values of the discrete Fourier transform of X. It had been conjectured that Williamson sequences exist for all orders n but the counterexample 35 was found in 1993.

Our Result

Williamson sequences were enumerated for odd orders up to 59 in 2007. Our work extends this enumeration to even orders up to 64. We do this by combining tools from the fields of satisfiability checking and symbolic computation.

SAT: Boolean Satisfiability Problem

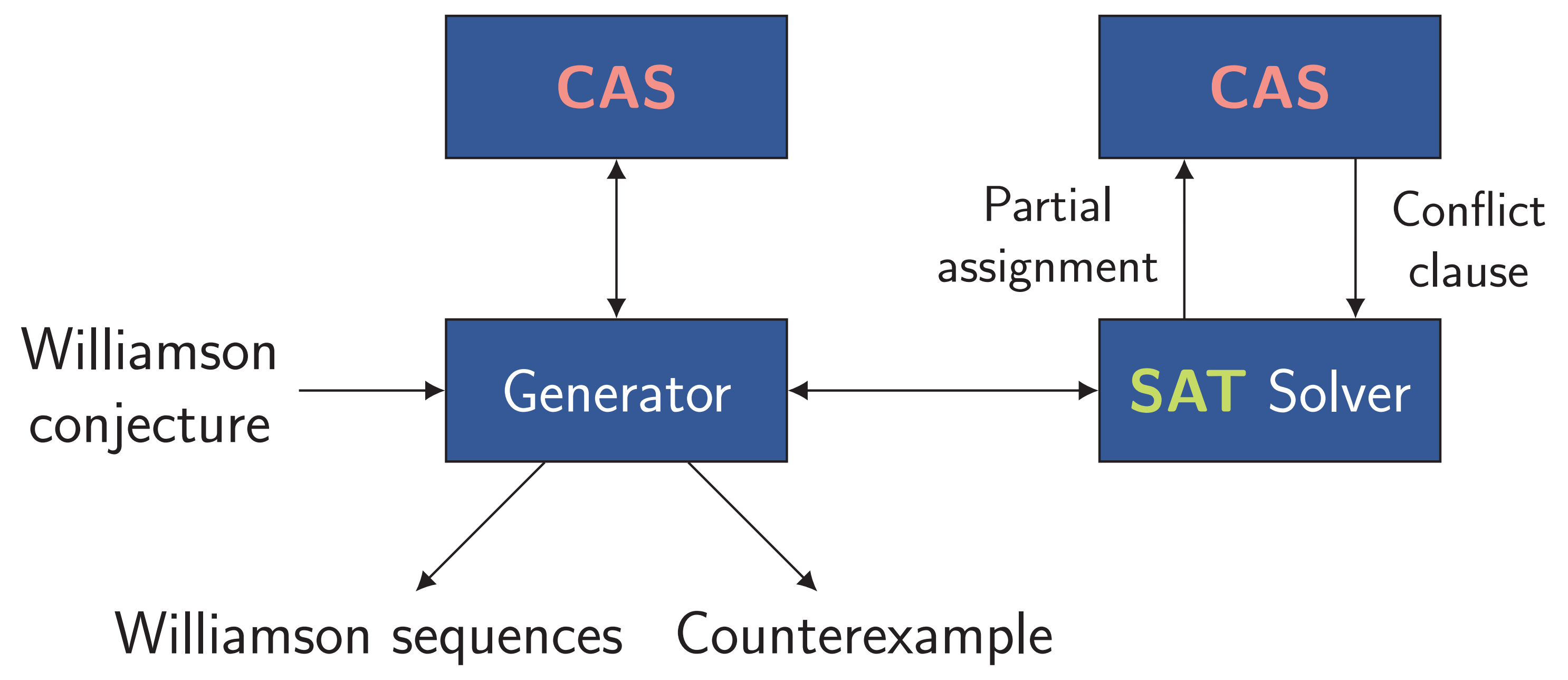
Many problems that can be rewritten as a SAT problem can be solved in practice using a modern SAT solver. SAT solvers contain efficient general-purpose search routines that perform a kind of “clever, inspired” brute force search.

CAS: Computer Algebra System

CAS contain functions that can efficiently solve a wide class of mathematical problems like computing the Fourier transform of a sequence.

Combining SAT+CAS

CAS are designed to solve domain-specific problems but do not generally use sophisticated general-purpose search. Our system uses the power of SAT solvers to perform an enumerative search along with CAS functions to compute intermediate quantities like PSD values.



Method Outline

A generator script splits the search space into subspaces and generates a SAT instance for each subspace. However, some constraints like $(*)$ cannot easily be encoded in a SAT instance. To work around this, we use a “programmatic” SAT solver that uses external CAS functions to encode such constraints.

Programmatic SAT

Periodically the SAT solver will make external calls to compute quantities like $\mathrm{PSD}_A(s)$. If this value is so large that $(*)$ cannot possibly be satisfied then a conflict clause is generated encoding that fact. The SAT solver will ignore assignments with sequence A in the future, greatly speeding up the search.

Conclusion

A programmatic SAT solver combined with CAS functionality is a powerful combination that can be useful when searching for large combinatorial objects. In our implementation a programmatic SAT+CAS solver could perform the search thousands of times more efficiently than an off-the-shelf SAT solver.