Effective Implementation of Hyper-Unit Propagation on FPGA and PC

Gábor Kusper and Krisztián Kusper

SAT solver algorithms are based on unit propagation. Unit propagation is a special case of hyper-unit propagation. Hyper-unit propagation is the propagation of an assignment. Usually hyper-unit propagation is implemented as a series of unit propagation by the units of the assignment. This means that it is not utilized that the units can be propagated simultaneously. We report a special literal matrix representation of SAT, which allows us to do hyper-unit propagation at once using only 3 operations per clauses. The 3 operation are 2 binary and operation and a comparison with zero. These operations are simple and allow us to implement hyper-unit propagation efficiently on a dedicated hardware using FPGA and on PC using low level programming languages like assembler. We compare several implementations.

If we use literal matrix representation of SAT we need at least two bits to represent a literal, because a literal is either positive or negative or the corresponding variable is not present (no occurrence literal). The basic idea of the special literal matrix is that we use the two bits as follows. If the first bit is set (1) and the second one is clear (0), it means the literal is positive. If the second bit is set and the first one is clear, it means the literal is negative. If both two bits are set or clear, it means the literal is the no occurrence literal. This is the new idea, because the usual literal matrix representations use only one combination of bits to represent the no occurrence literal.

We utilize this new idea as follows. First, we have to decide whether a clause becomes true or not if we assume, i.e., propagate an assignment. If they have a common literal then the clause becomes true. Second, we have to remove irrelevant literals form clauses that have no common literal with the assignments. This means we remove the negative of the assignment from these clauses. Both operations can be done using binary and operation, but in the first case we have to represent no occurrence literals in the assignment as 00. In the second case we have to use 11 to represent no occurrence literals in the assignment.

We implemented this technique on FPGA and on PC (32 bit x86-architecture).