MinkeyRink Solver at SMTCOMP 2020

Trevor Hansen
Melbourne, Australia

MinkeyRink Solver (MinkeyRink) is an eager bit-blasting (`QF_BV`) solver with a focus on applying simplifications that preserve and enhance sharing at the word-level (over `QF_BV` expressions). Sharing-aware transformations have long been applied in similar contexts, such as and-inverter graphs (AIGs) [1].

At word-level node-creation time: unsigned interval, signed interval, pointwise, constant-bit[3], and wrapped-interval[2] analysis is performed to identify expression that can be rewritten to simpler equivalent terms (e.g. constants), or to less expensive operations. The analyses compliment each other. For example:

- that $ite(a, 2, 5)$ cannot equal 3, is determined by pointwise analysis, but not via the other analyses.
- If interval analysis determines that an expression must be positive, then any arithmetic right shifts of that expression can be replaced by logical right shifts.
- If constant-bit analysis detemines that the bvand of two operands to an addition is zero, then the addition can be replaced by an bit-vector "or".

MinkeyRink: encodes into CNF via the and-inverter graph sub-package of ABC[4], for SAT solving uses CaDiCal, handles arbitrary-precision bit-vectors using Steffen Beyers library, and uses some components from the STP solver.

References

[1] Robert Brummayer and Armin Biere. Local two-level and-inverter graph minimization without blowup. In *Proceedings of the 2nd Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS06)*, 2006.

[2] Graeme Gange, Jorge Navas, Peter Schachte, Harald Søndergaard, and Peter J. Stuckey. Interval analysis and machine arithmetic: Why signedness ignorance is bliss. *ACM Transactions on Programming Languages and Systems*, 37(1):1:1–1:35, january 2015.

[3] Trevor Hansen. *A Constraint Solver and its Application to Machine-Code Test Generation.* PhD thesis, University of Melbourne, 2012.

[4] Alan Mishchenko, Satrajit Chatterjee, and Robert Brayton. DAG-aware AIG rewriting: A fresh look at combinational logic synthesis. In *Proceedings of the 43rd Annual Design Automation Conference*, DAC '06, pages 532–535, New York, NY, USA, 2006. ACM.