At Least Two Solutions

Norbert Manthey
nmanthey@conp-solutions.com
Dresden, Germany

***Abstract*—This document describes the ATLEASTTWO tool, as well as the benchmarks that have been submitted to the SAT competition 2021. The tool takes a CNF and encodes a new CNF which is satisfiable only if the initial CNF has at least two satisfying solutions. The difference of the two solutions can be restricted to input variables only, to support formulas that have been encoded with auxiliary variables.**

I. INTRODUCTION

When encoding problems, one might ask whether given problem has more than one solution. This property is for example useful when generating logic puzzles to know whether a generated puzzle has multiple solutions. A related property is to encode that a formula has exactly one solution. Similarly, the encoding could be extended to ask for at least two satisfying assignments, or at most k satisfying assignments. However, the result formula would grow quadratically with k. The current tool only supports asking for at least two solutions. Other extensions would be to encode that at least a given number of K assignments need to be different. In this case, the currently used *at-least-k* constraint – a clause – would have to be replaced by a properly encoded cardinality constraint.

II. ENCODING AT LEAST TWO SOLUTIONS

The input for the ATLEASTTWO tool is an uncompressed CNF file with the formulas F. This formula has n variables, and m clauses, where we assume all variables n to be present. As in CNF, variables are represented as integers, we will use math rules to describe transformations. The resulting formula is basically encoded by the following steps: (1) duplicate F into a formula G, but add the offset n to all variables to obtain fresh variables, (2) encode the equivalence for all variable pairs in F and G, respecting the offset, and (3) enforce that are least one equivalence from (2) is falsified. The three steps will be explained in more details next.

A. Duplicate Input Formula

As a first step, we duplicate all clauses in F. During this step, for all clauses C_i, we add the offset n to the variable representation of all literals l_i when creating the new clauses.

$$G := \bigwedge_{C_i \in F} \bigvee_{l_i \in C_i} l_i + n$$

B. Encode Solution Equality

To encode whether two variable assignments are equivalent, we need to introduce a new variable for each existing variable in F, or at least for the range of selected variables whose satisfying assignment should be different. Let e be the number of variables from 1 to e which should have at least one different assignment. As the default case, the number of variables to consider is the number of variables in the input, i.e. $e = n$.

$$E := \bigwedge_{i=1}^{e} a_i \leftrightarrow (f_i \leftrightarrow g_i)$$

,

where a_i are variables that do not occur in F nor in G. Note, the variables f_i and g_i again have the offset of n again, i.e. $g_i = f_i + n$.

C. Force Inqeuality of Solutions

When combining all formulas above, we obtain a formula that encodes the same formulas twice, and a sub-formula that indicates whether the assignments for both formulas are equal:

$$R' = F \wedge G \wedge E$$

.

A model for this formula can assign the same truth value for all variable pairs in F and G. To make sure, the models are different, we need to ensure that at least one pair of variables is not equal, by encoding an at-least-one constraint for the new auxiliary variables:

$$R = F \wedge G \wedge E \wedge \bigvee_{i=1}^{e} \neg a_i$$

.

Formula R is the resulting formula that will be generated by the tool.

D. Properties of the Generated Formula

First, we discuss the relationship between the satisfiability of the input formula F and the produced formula R. When the input formula F is unsatisfiable, the resulting formula R is also unsatisfiable.

Next, when F has exactly one model, the truth values of this model can be assigned to satisfy F. Formula G will can only be satisfied with the same set of assignments. Consequently, all variables a_i will be assigned to $\top$. In this case, the last subformula of R will be falsified.

In the final case, multiple model I and J exist to satisfy F. Then, I can satisfy F, and J can satisfy G by respecting the offset n. As I and J are diferent, at least one variable a_i will be assigned to $\bot$, resulting in a satisfying assignment for R. Note, such a satisfying assignment to R contains the two models I and J that satisfy the input formula F.

III. The Submitted Benchmark

The submitted benchmark formulas have been generated by using formulas from previous SAT competition benchmarks. The initial set of formulas to select the submitted formulas from is the robust benchmark for tuning SAT solvers from [1].

IV. Availability

The source of the tool is publicly available under the MIT license at https://github.com/conp-solutions/cnfmiter.

References

[1] H. H. Hoos, B. Kaufmann, T. Schaub, and M. Schneider, "Robust benchmark set selection for boolean constraint solvers," in *Revised Selected Papers of the 7th International Conference on Learning and Intelligent Optimization - Volume 7997*, ser. LION 7. Berlin, Heidelberg: Springer-Verlag, 2013, p. 138–152.