Logic Synthesis using SAT Solver Lingeling

Wilmer V. Uruchi Ticona

June 24, 2019

1 Introduction

The algorithm runs in iterations. It starts with a heap of size 1, if no solution is found, then it increases the size by 2, and it keeps increasing until a solution is found. For each iteration on heap size, the algorithm iterates through the maximum number of nor gates allowed for that size of the heap. It starts at nor gates less than 1, and keeps adding +1 until this number reaches the maximum number of nor gates allowed for the size of the heap ($floor(heap.size()/2)$). Inside the constructor of the CNF, there is a process that builds a constraint in CNF of the form $a_1 + a_2 + \ldots + a_n < k$ (naive n choose k), where k is the number of nor gates allowed (less than), and a represents the indicators (TRUE if the item is a nor gate, FALSE otherwise) for each item of the heap. The objective is to find the solution with the lowest depth and less number of nor gates.

2 Model

Input data is stored in $vars[numberVars+1][numberRows]$, the result for each comparison is stored in $result[numerRows]$.

The heap mentioned in the introduction is just an ordering principle for our binary tree, where the root starts with index 0, and we can find the children of any node using the formulas: $2 * i + 1$ for the left child, and $2 * i + 2$ for the right child, where i is the index of the item of the tree (heap) we are evaluating, except in the case of the leaves, the items that have no children. Also, we can find the parent of any item, except the root, by using $(i - 1)/2$.

Almost any item of the tree can be the constant 0, or any of the variables $x_1, x_2, \ldots, x_k$, or a NOR GATE; except the leaves, which cannot be NOR GATES.

Lets define n as the number of items in the heap, then we define $indicator(i, numIndicaPerNode-1)$ as a literal that is TRUE when the i-th item of the tree is a NOR GATE, FALSE otherwise, the other values in $indicator(i, 0..numberIndicatorsPerNode - 2)$ tell if the node is taking the value of the constant 0, or any variable. Let's define x as the matrix of $numberRows \times (v + 2)$ items, where its first column represents the results that come in

the input, the second column represents the constant 0, and subsequent columns represent the variables v.

Then we have *inner* as the matrix of size $numberRows \times v$ that represents the inner values that each item in the heap will take for comparison purpose. If, for example, a item in the heap assumes the value of the constant 0, then the corresponding *indicator* of the heap item will be TRUE, and the other indicators for that item will be 0.

When an item i of the heap assumes the role of a nor gate, the parents of that i-th item are taken as the input of the NOR GATE, the corresponding value is calculated and stored in the corresponding *inner*, the *indicator* signals that the item i is a NOR GATE.

There is no objective function. To replace that, the iteration described in the Introduction is performed.

From the solutions to those instances that could be compared with the results provided with the project statement, 248 have been optimally solved.