This is a guide for the use of the CircuitOptimization.

ABOUT:

You must specifying the number of inputs, which gate types you want, the cost of each operation, the maximum cost you want to search up to, the truth value you are looking for, and a directory. The program will run either until there are no more circuits with a cost less than the max cost, or until it finds the specified truth value it was told to look for. It will return a HashMap that maps truth values to an ArrayList of Circuits. The list of circuits that each truth value all have the same cost. This cost is the minimum cost of a circuit that can produce the truth value. The list contains all circuits with that cost that give the truth value. At certain checkpoints it will format this HashMap and save it to a file in the specified directory. The directory must contain “/” instead of “\” and it should end with “/” if it is the folder you want to save the file in. It should end with “.json” if it is the folder followed by the name of the file. If no file name is given (the directory doesn’t end with “.json”), it will generate a file name which may overwrite files in that directory if they happen to have the same name as the generated name. The generated file name will include the cost reached, the gates used, and the date. The program will also keep track of how long it took to arrive at the checkpoint when it saves the contents to a file.

NOTES:

* The program will overwrite files in the directory if it has the same name as the generated name if no name is specified, but this is unlikely because the date is included in the name. The programs saves a new file for each cost reached so it is recommended that you make a new folder and use that as the directory because multiple json files will be made.
* The program will return null if it is fed a set of allowed operations if all possible truth values for the number of inputs cannot be found with that set of allowed operations.
* The program will return null if you try to run it with inputs other than 2 or 3.
* The program will change the cost of anything that has a cost less than 0 to 1. If any operator besides the NOT, ~, has a cost of 0, it will change it to 1.
* Costs for circuits are rounded to two decimal places, but costs for operations are not.
* Setting the maxCost too high may lead to a memory error because too many things will be saved. Suggested setting is around 8 if all costs are 1.0, but depends on what the costs are and which operations are allowed. Including a cost of 0, you may want to lower the maxCost to around 5.
* If you just want it to run until the maxCost, input null for truthValueToFind.
* Adding operations that are not recognized will not affect the program. It will just remove them.
* If these are the values for a and b, the following table describes the operations that can be used to combine a and b. a=0011, b=0101

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Truth Value** | **Representation** |
| NOT | True when the input is False | 1100 | (~a) |
|  |  |  |  |
| AND | True when both a and b are true | 0001 | (a&b) |
| NAND | Opposite of AND; True when at least one is False | 1110 | (a@b) |
|  |  |  |  |
| OR | True when at least one is true | 0111 | (a+b) |
| XOR | True when only one is True not both | 0110 | (a^b) |
| NOR | Opposite of OR; True when neither are True | 1000 | (a.b) |
| XNOR | True if both are true or both are False | 1001 | (a=b) |
|  |  |  |  |
| IMPLIES | True if a is False or both a and b are True | 1101 | (a>b) |
| NIMPLIES | Opposite of IMPLIES | 0010 | (a$b) |

HOW TO:

1. Open the java file and navigate to ConstantProperties.java. There will be some example values already present. Leave the first two lines creating approvedOperators and costPerOp alone.
2. Decide whether you want two or three inputs.
3. Look at the allowed ops. Inside the parentheses enclosing them list the operators you want to include. Remove the ones you don’t want. Make sure operators are surrounded with double quotation marks and that they are separated by commas.
4. Set the maxCost to what you want it to be. This can be either a decimal or a whole number. It will be converted to a decimal automatically.
5. Leave truthValueToFind as null if you want to run it until it reaches the maxCost. If you want it to stop when it finds a specific truth value, replace the null with a truth value surrounded by double quotation marks.
6. Enter the directory in the format shown as dir.
7. Look at the lines that say costPerOp.put("~",1.0); in the setCostPerOp() method. Change the cost of the operations as necessary by replacing the 1.0 with the cost you want it to have. Make sure the number includes a decimal followed by a zero if it is a whole number.
8. Navigate to FindMinimalCircuit.java. Don’t change the line that creates answer, but you can add operations to perform on answer below it.
9. Click run. Your files will start saving in the specified directory.