How to simplify boolean formulas using mathematica.

In[4]:=
```
f = Or[h1 == l1 && h0 ≠ l1 && h1 == l0 && h0 ≠ l0 && h0 ≠ h1,
 h0 == l1 && h0 ≠ l0 && h0 == h1,
h1 == l1 && h0 ≠ l1 && h1 ≠ l0 && h0 ≠ l0 && h0 ≠ h1];

FindInstance[f, {h0, h1, l0, l1}, Booleans]
```

Out[5]=
```
{{h0 → True, h1 → False, l0 → False, l1 → False}}
```

```
BooleanMinimize[f]
```

Out[6]=
```
(h0 == h1 && h0 ≠ l0 && h0 == l1) || (h0 ≠ h1 && h0 ≠ l0 && h0 ≠ l1 && h1 == l1)
```

In[7]:=
```
FullSimplify [f]
```

Out[7]=
```
(h0 == h1 || h0 ≠ l1) && h0 ≠ l0 && (h0 == l1 || h1 == l1)
```

In[8]:=
```
f = Or[h1 == l1 && h0 == l0 && h0 ≠ h1, h1 == l1 && h0 == l0 && h0 == h1];
FullSimplify [f]
```

Out[9]=
```
h0 == l0 && h1 == l1
```

How to count using mathematica. The first cell counts a formula without projection. sharpSat reports an exact count after 37 seconds, where *Mathematica* run out of memory after a few minutes (process is killed by the kernel). The second cell counts a formula with projection. See the cnfToMath.py script for more details on how to convert from the DIMACS CNF file to a *Mathematica* expression.

In[1]:=
```
booleanExpr = << "~/workspace/bvcount/inputs/s-rsa-1.smt2 .cnf.m ";
(*booleanExpr = << "~/workspace/bvcount/inputs/test.m "*)
SatisfiabilityCount[booleanExpr] // AbsoluteTiming
```

In[62]:=
```
booleanExpr = << "~/workspace/bvcount/inputs/s-rsa-10.smt2 .cnf.m ";
projection = << "~/workspace/bvcount/inputs/s-rsa-10.smt2 .cnf.proj.m ";
SatisfiabilityCount[booleanExpr, projection] // AbsoluteTiming
```

Out[64]=
```
{0.042996,
  SatisfiabilityCount[ ··· 1 ··· , {v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13,
      v14, v15, v16, v17, v18, v19, v20, ··· 24 ··· , v45, v46, v47, v48, v49, v50,
      v51, v52, v53, v54, v55, v56, v57, v58, v59, v60, v61, v62, v63, v64}]}
```

large output | show less | show more | show all | set size limit ...

As you can see, SatisfiabilityCount doesn't work when we specify a projection set distinct from the set of all boolean variables in the formula.

In[65]:=
```
SatisfiabilityCount[Or[x && y && z, x || y], {x, y}]
```

Out[65]=
```
SatisfiabilityCount[(x&&y&&z)||x||y, {x, y}]
```

In[66]:=
```
SatisfiabilityCount[Or[x && y && z, x || y], {x, y, z}]
```

Out[66]= 6