

This notebook implements Algorithm 1 from “[Contextuality without access to a tomographically complete set](#)” by Matthew F. Pusey, LÍdia del Rio and Bettina Meyer. If you want to use the CDD interface to do vertex enumeration, you should first run [cdd\\_interface.nb](#).

© 2021 Matthew F. Pusey

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

(\* As an example, generate some stats for a qubit \*)

`n = 5;`

`η = 0.89;`

`points = Table[{η Sin[θ], η Cos[θ]} /. θ → 2 π  $\frac{x + 1/4}{n}$ ,`

`{x, 0, n - 1}]; (* Bloch vectors of n states`

`(X and Z components only) *)`

`meas = {{0, 1}, {1, 0}, {Sin[3 π/10], Cos[3 π/10]}};`

(\* Bloch vectors of measurement projectors \*)

`Show[RegionPlot[And @@ (-1 ≤ #.{x, y} ≤ 1 & /@ meas),`

`{x, -1.2, 1.2}, {y, -1.2, 1.2}],`

`Graphics[{{Gray, Circle[]},`

`MapIndexed[Text[# 2, # 1] &, points]]]`

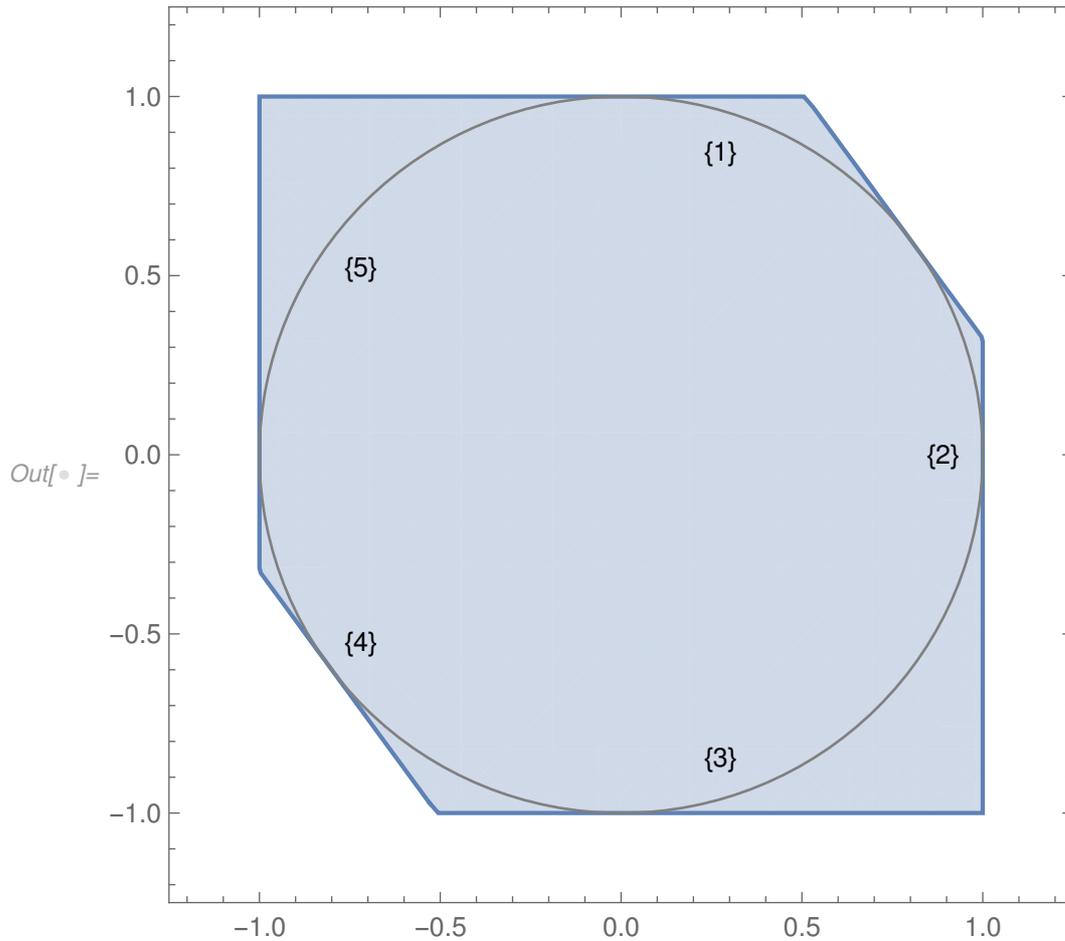
(\* Show the polygon where the measurements in meas would give valid probabilities,

the Bloch sphere, and the states as numbers \*)

`stats = Table[ $\frac{r.p + 1}{2}$ , {p, points}, {r, meas}];`

(\* Calculate the probability of each measurement outcome for each state \*)

`Labeled[MatrixForm[stats], "stats =", Left]`



Out[•]= stats =

$$\begin{pmatrix} 0.92322 & 0.637513 & 0.860013 \\ 0.5 & 0.945 & 0.860013 \\ 0.0767799 & 0.637513 & 0.362487 \\ 0.238436 & 0.139987 & 0.055 \\ 0.761564 & 0.139987 & 0.362487 \end{pmatrix}$$

(\* From now on we just need the matrix stats,  
could come from above calculation,  
or from experimental data, etc\*)

`{n, m} = Dimensions[stats];`

`λs = 2m; (* The hidden variable gives a  
deterministic outcome to every measurement,  
so there are 2m possible values *)`

`λfor[meas_] := Table[If[λ[[meas]] == 1, 1, 0],  
{λ, Tuples[{0, 1}, m]}]`

(\* Gives a vector of length λs,

where each entry is 1 if the corresponding hidden variable says that measurement number  $i$  occurs, and 0 otherwise \*)

(\* N.B.: CDD is given (in)equalities on points in the polytope  $x$  in the form of a vector  $a$ , so that  $a_0 + a_1x_1 + a_2x_2 + \dots$  is  $\geq 0$  or  $= 0$  \*)  
`norm = Join[{-1}, ConstantArray[1,  $\lambda$ s]];`  
 (\* Normalization equality:  $-1 + \sum_{\lambda} \text{prob}(\lambda) = 0$  \*)  
`pos = Join[{0},  $\#$ ] &/@ IdentityMatrix[ $\lambda$ s];`  
 (\* Positivity inequality: for each  $\lambda$ ,  $\text{prob}(\lambda) \geq 0$  \*)

`reproduce[stat_] :=`

`MapIndexed[Join[{- $\#$  1},  $\lambda$ for[ $\#$  1]]] &, stat]`

(\* Equality of reproducing the operational statistics in stat: for each measurement  $i$ ,  $-\text{stat}_i + \sum_{\lambda \in \lambda_{\text{for}(i)}} \text{prob}(\lambda) = 0$  \*)

(\* N.B.: `cddHtoV` takes two arguments: a list of equalities and a list of inequalities. It returns a list of extreme points \*)

`poly[stat_] := cddHtoV[Join[{norm}, reproduce[stat]], pos]`

(\* Use CDD to find extreme points of polytope of hidden-variable distributions that reproduce stat \*)

(\* Consider two polytopes with lists of vertices  $v_1$  and  $v_2$

If they do not intersect, then there exists a gap between them, i.e. there exists  $x$ ,  $c_1$ ,  $c_2$  with  $v \cdot x \leq c_1$  for all  $v \in v_1$  whereas  $v \cdot x \geq c_2$  for all  $v \in v_2$ , and  $c_1 < c_2$ .

Here this is implemented as a linear program: the first variable is  $c_1 - c_2$ , which is minimized  
 The second variable is  $c_1$   
 The remaining variables are  $x$   
 To ensure the problem always has a bounded solution we require the first variable is  $\geq -1$   
 If the polytopes do intersect, then  $c_1 \geq c_2$  and so the first variable is positive. In this case the minimum value is 0, because we can always set  $c_1$ ,  $c_2$  and  $x$  to zero.

\*)

```

intersection[v1_ , v2_ ] := Module[{one1, one2},
  one1 = ConstantArray[1, {Length[v1], 1}];
  one2 = ConstantArray[1, {Length[v2], 1}];
  LinearProgramming[Join[{1}, ConstantArray[0,  $\lambda$ s + 1]],
    ArrayFlatten[ $\begin{bmatrix} 0 & \text{one1} & \text{one1} & -v1 \\ \text{one2} & -\text{one2} & v2 \end{bmatrix}$ , Flatten[ $\begin{bmatrix} 0 & \text{one1} \\ \text{one2} \end{bmatrix}$ ,
    Join[{-1}, ConstantArray[- $\infty$ ,  $\lambda$ s + 1]]]]
intersects[v1_ , v2_ ] := Chop[intersection[v1, v2][[1]]] == 0

```

```

In[ ] := (* Find all pairs of disjoint subsets of states
with size at least 2 *)

```

```

pointidxs = Range[n];
pairidxs = DeleteDuplicates[
  Flatten[
    Table[Sort[{x, #}] & /@
      Subsets[Complement[pointidxs, x], {2, n}],
      {x, Subsets[pointidxs, {2, n - 2}]}], 1];

```

```

In[ ] := (* Find the polytope for each state using cdd *)
polys = poly /@ stats;

```

```
In[• ]:= (* For each pair of subsets of states,
           figure out if the convex hull of their polytopes
           intersect. If they intersect,
           then there exists a mixture of each subset
           that can be represented by the same distribution
           over hidden variables. For all we know,
           this might be the only mixture that gives the
           same operational probabilities for the unknown
           additional measurements. Hence we cannot
           prove preparation contextuality. So we're
           hoping to get all False. *)
```

```
result =
intersects[Join @@ polys[[# 1],
                  Join @@ polys[[# 2]] & @@@ pairidxs
If[Or @@ result,
  "I could not determine whether there is a
  noncontextual model",
  "The proof of contextuality is robust"]
```

```
Out[• ]= {False, False, False, False, False, False,
          False, False, False, False, False, False,
          False, False, False, False, False, False,
          False, False, False, False, False, False, False}
```

```
Out[• ]= The proof of contextuality is robust
```