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.

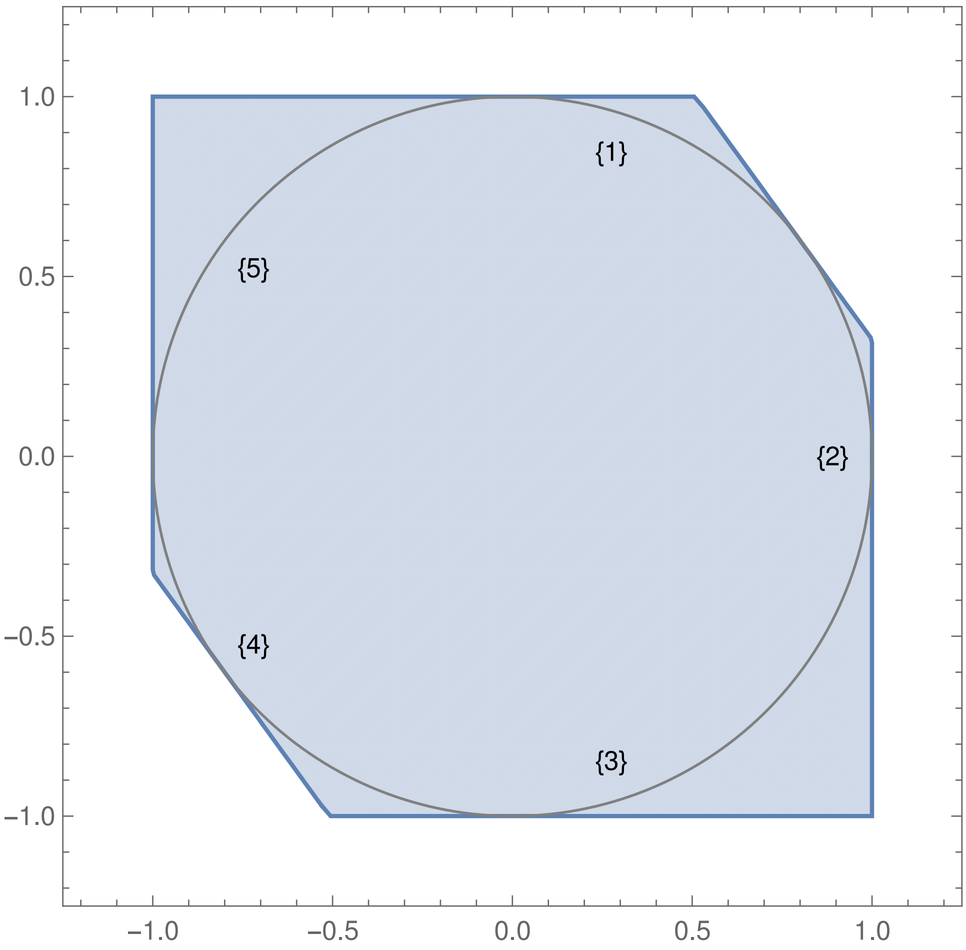
```
(* As an example, generate some stats for a qubit *)
n = 5;
η = 0.89;
points = Table[{η Sin[θ], η Cos[θ]} /. θ → 2 π (x + 1/4)/n,
  {x, 0, n - 1}];(* Bloch vectors of n states
 (X and Z components only) *)

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[(r.p + 1)/2, {p, points}, {r, meas}];
(* Calculate the probability of each measurement
 outcome for each state *)
Labeled[MatrixForm[stats], "stats =", Left]
```

Out[•]=



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 = 2^m; (* The hidden variable gives a
 deterministic outcome to every measurement,
so there are 2^m 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
 meas 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_1 x_1 + a_2 x_2 + ... is ≥0 or =0 *)
norm = Join[{-1}, ConstantArray[1, λs]];
(* Normalization equality: -1 + ∑_λ prob(λ) = 0 *)
pos = Join[{0}, #] & /@ IdentityMatrix[λs];
(* Positivity inequality: for each λ, prob(λ) ≥ 0 *)
reproduce[stat_] :=
 MapIndexed[Join[{-#1}, λfor[#2[[1]]]] &, stat]
(* Equality of reproducing the operational
 statistics in stat: for each measurement i,
-stat_i + ∑_{λ ∈ λfor(i)} prob(λ) = 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
 v1 and v2
 If they do not intersect,
then there exists a gap between them,
i.e. there exists x, c1,
c2 with v.x ≤ c1 for all v ∈
 v1 whereas v.x ≥ c2 for all v ∈ v2, and c1 < c2.
```

```
   Here this is implemented as a linear
  program: the first variable is c1 - c2,
which is minimized
  The second variable is c1
  The remaining variables are x
  To ensure the problem always has a bounded
  solutino we require the first variable is ≥ -1
  If the polytopes do intersect,
then c1 ≥ c2 and so the first variable is
  positive. In this case the minimum value is 0,
because we can always set c1, c2 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, λs + 1]],
   ArrayFlatten[{{0, one1, one1, -v1}, {one2, -one2, v2}}], Flatten[{{0, one1}, {one2}}],
   Join[{-1}, ConstantArray[-∞, λs + 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
```