Milk problem - a graphs solution

GONZALO GARCÍA ALARCÓN ESTRADA, **********
Nov / 16 / 2017
University of Toronto, Shameless Mathematica

Milk problem:

A milkman has an 8 liter jar, a 5 liter jar and 3 liter jar. The 8 liter jar is full of milk. Can the milkman measure out 4 liters? He has no other way to measure besides using these jars.

Challenge:

Draw the state graph of this problem (no spilling allowed!)

```
S = Solve[(x + y + z == 8) && (8 ≥ x ≥ 0) ∧ (5 ≥ y ≥ 0) ∧ (3 ≥ z ≥ 0), {x, y, z}, Integers];

States = {x, y, z} /. S
```

```
{{0, 5, 3}, {1, 4, 3}, {1, 5, 2}, {2, 3, 3}, {2, 4, 2}, {2, 5, 1}, {3, 2, 3}, {3, 3, 2},
 {3, 4, 1}, {3, 5, 0}, {4, 1, 3}, {4, 2, 2}, {4, 3, 1}, {4, 4, 0}, {5, 0, 3}, {5, 1, 2},
 {5, 2, 1}, {5, 3, 0}, {6, 0, 2}, {6, 1, 1}, {6, 2, 0}, {7, 0, 1}, {7, 1, 0}, {8, 0, 0}}
```

```
(*compare .... -1 out, 0 equal, +1 in*)
lzg[x_, y_] := If[x > y, -1,
  If[x == y, 0,
   If[x < y, +1]]]

(* now for sets.... actually lists *)
lzgSet[A_, B_] := Map[(lzg[A[[#]], B[[#]]]) &, Range[Min[Length[A], Length[B]]]]

(* RELATION A→B i.e. B is pourable from A *)
Pourable[A_, B_] := Module[{change, full = {8, 5, 3}},
  change = lzgSet[A, B];
  If[MemberQ[change, 0],
   If[(Length[Position[change, -1]] == 1) ∧ (Length[Position[change, +1]] == 1),
    out = Position[change, -1][[1]][[1]]; in = Position[change, +1][[1]][[1]];
    If[B[[out]] == 0 ∨ B[[in]] == full[[in]], True, False], False]]]


(*tests*)
a = {5, 3, 0}; b = {3, 5, 0};

Pourable[a, b]
```

```
True
```

```
Pourable[b, a]
```

```
False
```

```
(* apply relation to states *)
SxS = Pair @@@ Tuples[States, 2];
SxS2 = SxS /. Pair[i_, j_] :> i ↦ j /; Pourable[i, j];
SxS3 = Select[SxS2, Head[#] == DirectedEdge &]
```

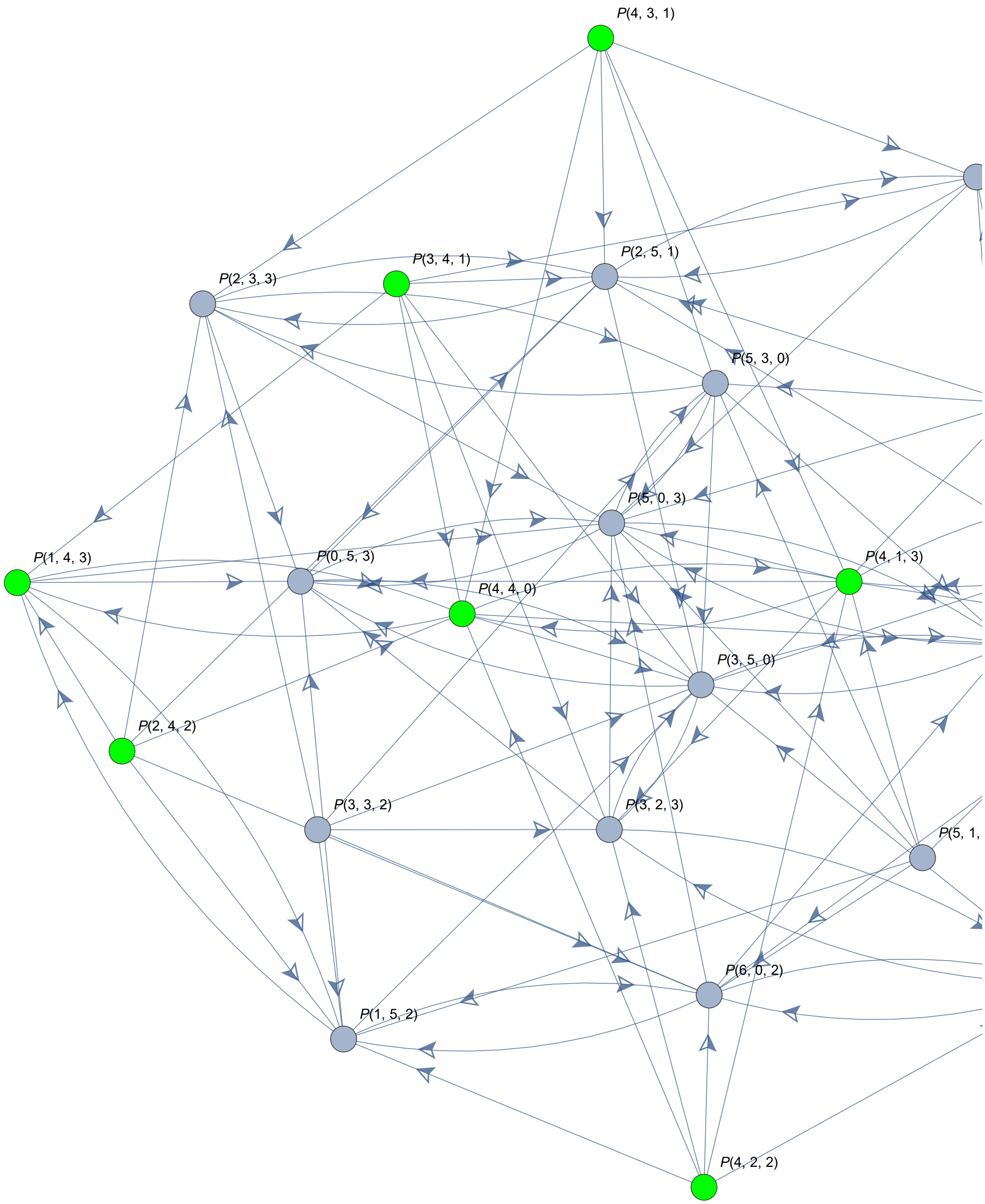
```
{{0, 5, 3} ↦ {3, 5, 0}, {0, 5, 3} ↦ {5, 0, 3}, {1, 4, 3} ↦ {0, 5, 3},
 {1, 4, 3} ↦ {1, 5, 2}, {1, 4, 3} ↦ {4, 4, 0}, {1, 4, 3} ↦ {5, 0, 3},
 {1, 5, 2} ↦ {0, 5, 3}, {1, 5, 2} ↦ {1, 4, 3}, {1, 5, 2} ↦ {3, 5, 0}, {1, 5, 2} ↦ {6, 0, 2},
 {2, 3, 3} ↦ {0, 5, 3}, {2, 3, 3} ↦ {2, 5, 1}, {2, 3, 3} ↦ {5, 0, 3}, {2, 3, 3} ↦ {5, 3, 0},
 {2, 4, 2} ↦ {1, 4, 3}, {2, 4, 2} ↦ {1, 5, 2}, {2, 4, 2} ↦ {2, 3, 3}, {2, 4, 2} ↦ {2, 5, 1},
 {2, 4, 2} ↦ {4, 4, 0}, {2, 4, 2} ↦ {6, 0, 2}, {2, 5, 1} ↦ {0, 5, 3}, {2, 5, 1} ↦ {2, 3, 3},
 {2, 5, 1} ↦ {3, 5, 0}, {2, 5, 1} ↦ {7, 0, 1}, {3, 2, 3} ↦ {0, 5, 3}, {3, 2, 3} ↦ {3, 5, 0},
 {3, 2, 3} ↦ {5, 0, 3}, {3, 2, 3} ↦ {6, 2, 0}, {3, 3, 2} ↦ {1, 5, 2}, {3, 3, 2} ↦ {2, 3, 3},
 {3, 3, 2} ↦ {3, 2, 3}, {3, 3, 2} ↦ {3, 5, 0}, {3, 3, 2} ↦ {5, 3, 0}, {3, 3, 2} ↦ {6, 0, 2},
 {3, 4, 1} ↦ {1, 4, 3}, {3, 4, 1} ↦ {2, 5, 1}, {3, 4, 1} ↦ {3, 2, 3}, {3, 4, 1} ↦ {3, 5, 0},
 {3, 4, 1} ↦ {4, 4, 0}, {3, 4, 1} ↦ {7, 0, 1}, {3, 5, 0} ↦ {0, 5, 3}, {3, 5, 0} ↦ {3, 2, 3},
 {3, 5, 0} ↦ {8, 0, 0}, {4, 1, 3} ↦ {0, 5, 3}, {4, 1, 3} ↦ {4, 4, 0}, {4, 1, 3} ↦ {5, 0, 3},
 {4, 1, 3} ↦ {7, 1, 0}, {4, 2, 2} ↦ {1, 5, 2}, {4, 2, 2} ↦ {3, 2, 3}, {4, 2, 2} ↦ {4, 1, 3},
 {4, 2, 2} ↦ {4, 4, 0}, {4, 2, 2} ↦ {6, 0, 2}, {4, 2, 2} ↦ {6, 2, 0}, {4, 3, 1} ↦ {2, 3, 3},
 {4, 3, 1} ↦ {2, 5, 1}, {4, 3, 1} ↦ {4, 1, 3}, {4, 3, 1} ↦ {4, 4, 0}, {4, 3, 1} ↦ {5, 3, 0},
 {4, 3, 1} ↦ {7, 0, 1}, {4, 4, 0} ↦ {1, 4, 3}, {4, 4, 0} ↦ {3, 5, 0}, {4, 4, 0} ↦ {4, 1, 3},
 {4, 4, 0} ↦ {8, 0, 0}, {5, 0, 3} ↦ {0, 5, 3}, {5, 0, 3} ↦ {5, 3, 0}, {5, 0, 3} ↦ {8, 0, 0},
 {5, 1, 2} ↦ {1, 5, 2}, {5, 1, 2} ↦ {4, 1, 3}, {5, 1, 2} ↦ {5, 0, 3}, {5, 1, 2} ↦ {5, 3, 0},
 {5, 1, 2} ↦ {6, 0, 2}, {5, 1, 2} ↦ {7, 1, 0}, {5, 2, 1} ↦ {2, 5, 1}, {5, 2, 1} ↦ {3, 2, 3},
 {5, 2, 1} ↦ {5, 0, 3}, {5, 2, 1} ↦ {5, 3, 0}, {5, 2, 1} ↦ {6, 2, 0}, {5, 2, 1} ↦ {7, 0, 1},
 {5, 3, 0} ↦ {2, 3, 3}, {5, 3, 0} ↦ {3, 5, 0}, {5, 3, 0} ↦ {5, 0, 3}, {5, 3, 0} ↦ {8, 0, 0},
 {6, 0, 2} ↦ {1, 5, 2}, {6, 0, 2} ↦ {5, 0, 3}, {6, 0, 2} ↦ {6, 2, 0}, {6, 0, 2} ↦ {8, 0, 0},
 {6, 1, 1} ↦ {2, 5, 1}, {6, 1, 1} ↦ {4, 1, 3}, {6, 1, 1} ↦ {6, 0, 2}, {6, 1, 1} ↦ {6, 2, 0},
 {6, 1, 1} ↦ {7, 0, 1}, {6, 1, 1} ↦ {7, 1, 0}, {6, 2, 0} ↦ {3, 2, 3}, {6, 2, 0} ↦ {3, 5, 0},
 {6, 2, 0} ↦ {6, 0, 2}, {6, 2, 0} ↦ {8, 0, 0}, {7, 0, 1} ↦ {2, 5, 1}, {7, 0, 1} ↦ {5, 0, 3},
 {7, 0, 1} ↦ {7, 1, 0}, {7, 0, 1} ↦ {8, 0, 0}, {7, 1, 0} ↦ {3, 5, 0}, {7, 1, 0} ↦ {4, 1, 3},
 {7, 1, 0} ↦ {7, 0, 1}, {7, 1, 0} ↦ {8, 0, 0}, {8, 0, 0} ↦ {3, 5, 0}, {8, 0, 0} ↦ {5, 0, 3}}
```

```
(* ----- PREPARATION FOR GRAPH ------ *)

ver = P @@@ States;

edg = SxS3 /. a_ ↦ b_ :> P @@ a ↦ P @@ b;


Start = {P[8, 0, 0]};

Obj = Select[ver, MemberQ[List @@ #, 4] &]
```

```
{P[1, 4, 3], P[2, 4, 2], P[3, 4, 1], P[4, 1, 3], P[4, 2, 2], P[4, 3, 1], P[4, 4, 0]}
```

```
G = Graph[ver, edg, VertexLabels → "Name", ImageSize → 900,
    EdgeShapeFunction → GraphElementData["ShortCarvedArrow", "ArrowSize" → 0.02]];
```

```
HighlightGraph[G, {Style[Start, Orange], Style[Obj, Green]}]
```



```
(* find reachable objectives *)
reachPaths = Map[FindShortestPath[G, P[8, 0, 0], #] &, Obj];

NotObj = Select[Obj, ! MemberQ[Flatten[reachPaths], #] &]
YesObj = Complement[Obj, NotObj]
```

```
{P[2, 4, 2], P[3, 4, 1], P[4, 2, 2], P[4, 3, 1]}
```

```
{P[1, 4, 3], P[4, 1, 3], P[4, 4, 0]}
```

```
HighlightGraph[G, {Style[Start, Blue], Style[YesObj, Green], Style[NotObj, LightGreen]} ⋃
  Map[PathGraph[#, DirectedEdges → True] &, reachPaths]]
```