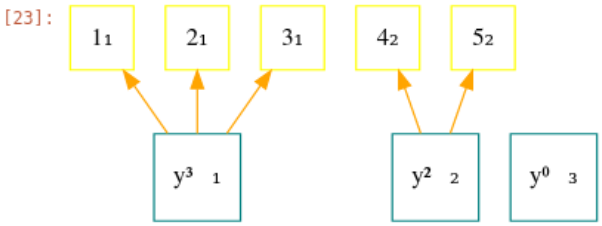
Exploring Poly through Catlab's Poly.jl

Lauresa Stilling

Working together with Thomas using CatLab's Poly.jl codebase and GraphViz we implemented a corolla esc representation of a FinPoly object. We explored how to implement a mapping between p → q where p and q are two FinPoly objects. The mapping requires a user to specify for each position of p a corresponding position in q and for each direction in q based upon the position of p a mapping back to directions in p. The ◁ tri operator (product composition) has been implemented between two FinPoly objects.

```
[23]: 1 GraphPoly(FinPoly([3,2,0]))
```



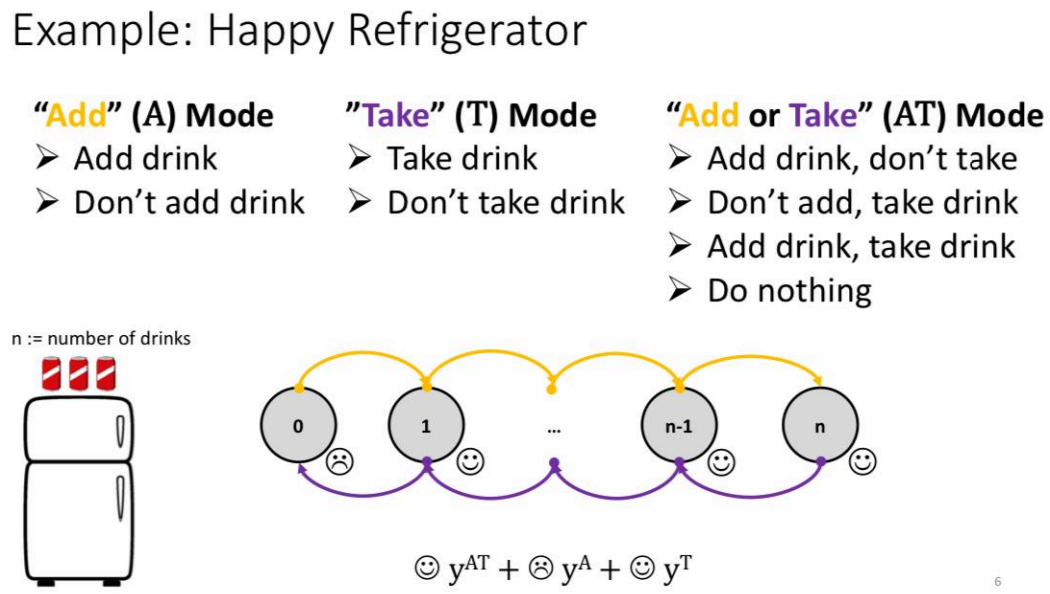
Corolla esc representation of $y^3 + y^2 + 1$

Creating a Moore Machine based upon set States, a Polynomial and and a Poly Map

Using the “Polynomial functors in Catlab” slideshow by Angeline Aguinaldo, , Kris Brown, and Marco Perin from ACT Conference 2021 I explored how to translate a Sy^S —> p model to a Finite State Automata (FSA) representation of the system based upon a set S number of states (n), and a mapping between states to a polynomial. By setting S = n, and creating n nodes, where each node is named based upon the output in p that it would be mapped to. For each state the transitions between the states are based upon the directions of the polynomial back to the updated state it would map back to. Either labeling the arrows by the original position and the updated position or by context of the system described (example of the latter below). To identify the initial state one can double circle it.

Example: “Happy Refrigerator"

Within the slide show example we have a refrigerator that based upon the contents of the refrigerator will allow a user to do varying actions. Described below are the “Modes” (positions) of the refrigerator; “Add”, “Take”, “Add or Take”, based on these modes we have corresponding behaviours (directions) that will impact the state of our refrigerator the behaviours are “Add drink” (+), “Don’t add drink” (0), “Take drink” (-), “Don’t take drink” (0), “Add drink, don’t take” (+), “Don’t add, take drink” (-), “Add drink, take drink” (swp), and “Do nothing” (0).



Example: Happy Refrigerator

```
D = PolyDynam(p, # polynomial
              5, # number of states 🥫🥫🥫🥫
              [2, 1, 1, 1, 3], # position per state
              [add, add_and_take, add_and_take, add_and_take, take], # behavior
              per state
              2) # initial state index
```

The polynomial is: 🙂y^{AT} + 😭y^{A} + 🙂y^{T} OR 🙂y^{4} + 😭y^{2} + 🙂y^{2}

In PolyDynam the presenters create an instance of the dynamical system having 5 states.

The describing the position mapping for each state as an array

φ_1: Initial Position |--> Position in Polynomial

1 |--> 2, 2 |--> 1, 3 |--> 1, 4 |--> 1, 5 |--> 3

$\varphi_i^\sharp$: Initial Position: Direction in p |--> New State in System

1: + |--> 2
0 |--> 1

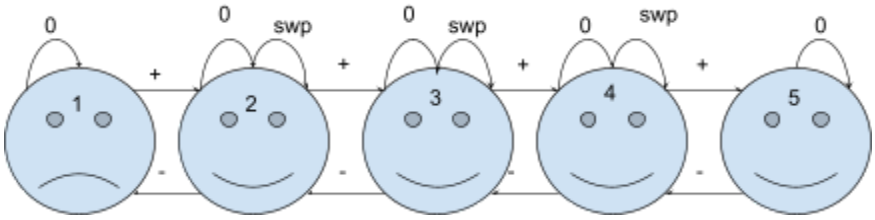
2: + |--> 3
- |--> 1
swp |--> 2
0 |--> 2

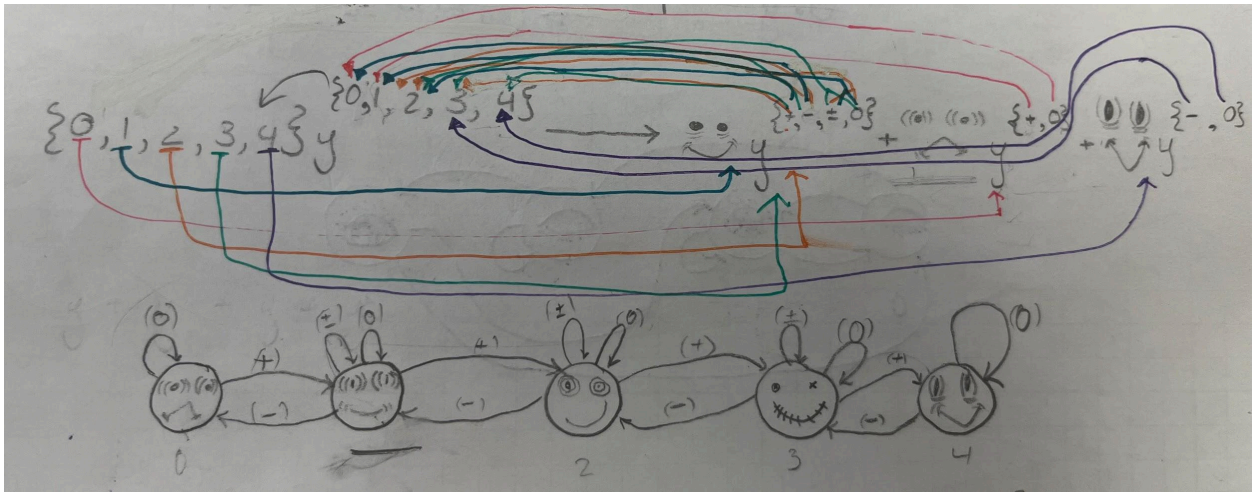
3: + |--> 4
- |--> 2
swp |--> 3
0 |--> 3

4: + |--> 5
- |--> 3
swp |--> 4
0 |--> 4

5: - |-->4
0 |--> 5

The machine represents what the states will be based upon the chosen direction.




{0,1,2,3,4} y
{0,1,2,3,4}
{+,-,±,0}
{+,0}
{-,0}
y
y
y
y
1
+
+
(0)
(±)
(0)
(±)
(0)
(±)
(0)
(0)
(+)
(+)
(+)
(+)
(−)
(−)
(−)
(−)
0
2
3
4