example1

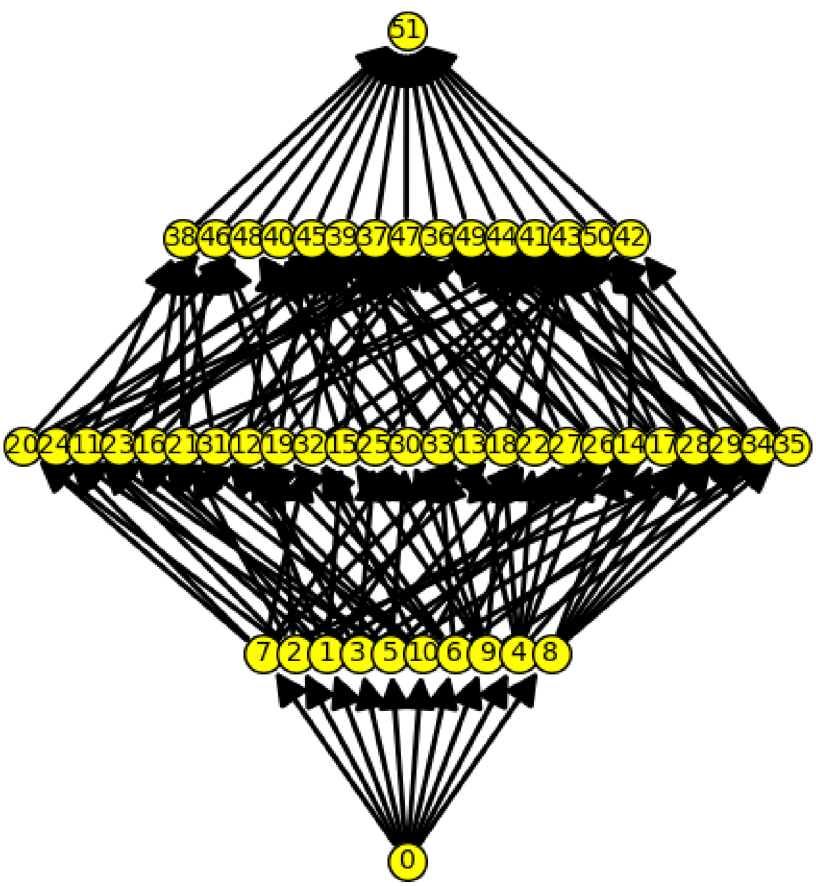
September 19, 2024

1 Chow Polynomial of Braid-5

First, we initialize the matroid Braid-5.

[7]:
```
n = 5
edgelist = sorted(graphs.CompleteGraph(n).edges(labels=False))
matroid = Matroid(graph=edgelist, groundset=edgelist)
flats = [list(matroid.flats(i)) for i in range(n)]
all_elements = sum(flats, [])  # Flatten the list of flats
labels = {element: idx for idx, element in enumerate(all_elements)}
matroid.lattice_of_flats().plot(
    element_labels = labels, element_color = "yellow")
```

[7]:



We now generate the possible degrees that a flat can have in a monomial. For example, if $[x_1, x_2, x_3, x_4]$ is a chain and $[0, 1, 0, 2]$ is a weight, then we say that $x_2 x_4^2$ is an FY-monomial.

[8]:
```
def generate_weights(rank):
    weights = set()
    for i in range(1, rank):
        for j in range(rank):
            weight = [0] * rank

            if i >= j:
                weight[i] = j
                weights.add(tuple(weight))

            if rank - (i+1) > 1:
                # Recursion to get the complete list of weights...
                y = generate_weights(rank - (i + 1))
                for x in y:
                    temp_weight = weight.copy()
                    weights.add(tuple(temp_weight[:i+1] + x))

    return [list(w) for w in weights]

rank = matroid.rank()
weights = generate_weights(rank)
```

Next, we compute all the possible FY-monomials in the example matroid.

[9]:
```
rflats = flats[1:] #empty flat is not required.
fy_monomials_list = [[] for _ in range(rank)]

def generate_monomials(weight, flats):
    if sum(weight) == 0: return [[]]
    # Find the first non-zero weight
    start_index = next(i for i in range(len(weight)) if weight[i] != 0)
    result = []

    for initial_flat in flats[start_index]:
        initial_monomial = [initial_flat] * weight[start_index]
        potential_combinations = [initial_monomial]

        for i in range(start_index + 1, len(weight)):
            if weight[i] != 0:
                new_combinations = []
                for flat in flats[i]:
                    for combo in potential_combinations:
```

```
                        if combo[-1].issubset(flat):
                            new_combo = combo + [flat] * weight[i]
                            new_combinations.append(new_combo)
                potential_combinations = new_combinations

        result.extend(potential_combinations)

    return result

for weight in weights:
    fy_monomials_list[sum(weight)].extend(generate_monomials(weight, rflats))

#example fy-monomial
print(fy_monomials_list[1][0])
```

```
[frozenset({(0, 1), (2, 4), (1, 2), (0, 4), (3, 4), (0, 3), (1, 4), (2, 3), (0,
2), (1, 3)})]
```

We write some simplification functions to make fy-monomials look simpler:

[10]:
```
def simplify(monomial):
    return tuple([labels[x] for x in monomial])

def set_simplify(monomial_set):
    return set(sorted([simplify(x) for x in monomial_set]))

for i in range(rank):
    print("\nrank: " + str(i))
    print(set_simplify(fy_monomials_list[i]))
```

```

rank: 0
{()}

rank: 1
{(51,), (11,), (14,), (17,), (23,), (20,), (26,), (29,), (35,), (32,), (38,),
(44,), (41,), (47,), (50,), (16,), (13,), (19,), (25,), (22,), (28,), (31,),
(37,), (34,), (40,), (43,), (49,), (46,), (12,), (18,), (15,), (21,), (24,),
(30,), (27,), (33,), (36,), (42,), (39,), (45,), (48,)}

rank: 2
{(43, 43), (38, 38), (24, 51), (11, 51), (33, 51), (35, 51), (39, 39), (15, 51),
(26, 51), (13, 51), (48, 48), (19, 51), (30, 51), (28, 51), (17, 51), (44, 44),
(32, 51), (40, 40), (49, 49), (36, 36), (34, 51), (21, 51), (45, 45), (14, 51),
(12, 51), (23, 51), (27, 51), (41, 41), (50, 50), (51, 51), (46, 46), (25, 51),
(29, 51), (16, 51), (20, 51), (18, 51), (31, 51), (22, 51), (37, 37), (47, 47),
(42, 42)}
```

```
rank: 3
{(51, 51, 51)}
```

The symmetric group $G = S_5$ acts on the example matroid. We now set up the appropriate functions required to compute the actions on the vertex set and obtain the stabilizer groups and the set of orbits.

[11]:
```
G = SymmetricGroup(range(n))

def action_on_flats(g, m):
    def action_on_groundset(g, x):
        return tuple(sorted(g(y) for y in x))
    return frozenset(sorted([action_on_groundset(g,x) for x in m]))

def action_on_fymonomials(g, monomial):
    return tuple(sorted([action_on_flats(g,m) for m in monomial]))

def stab(G, m, action):
    return G.subgroup(set(g for g in G if action(g, m) == tuple(m)))

def orbit(G, m, action):
    return frozenset(sorted(action(g, m) for g in G))

def orbits(G, X, action):
    return set(orbit(G, x, action) for x in X)
```

Finally, we compute the orbits of the fy-monomials under the action of G.

[12]:
```
for i in range(rank):
    set_of_orbits = orbits(G, fy_monomials_list[i], action_on_fymonomials)
    print("\nrank: " + str(i))
    for x in set_of_orbits:
        print(set_simplify(x))

#print(fy_monomials_list[2][0])
#print(orbits(G, fy_monomials_list[2], action_on_fymonomials)[1])
```

```
rank: 0
{()}

rank: 1
{(41,), (47,), (40,), (49,), (46,), (42,), (45,), (48,), (38,), (44,)}
{(28,), (31,), (15,), (21,), (34,), (24,), (14,), (27,), (33,), (23,), (20,),
(26,), (16,), (19,), (25,)}
{(12,), (18,), (35,), (11,), (30,), (17,), (29,), (13,), (32,), (22,)}
{(51,)}
{(37,), (50,), (43,), (36,), (39,)}
```

```
rank: 2
{(51, 51)}
{(46, 46), (45, 45), (49, 49), (38, 38), (44, 44), (48, 48), (41, 41), (47, 47),
(40, 40), (42, 42)}
{(29, 51), (35, 51), (11, 51), (30, 51), (17, 51), (12, 51), (18, 51), (13, 51),
(32, 51), (22, 51)}
{(25, 51), (34, 51), (21, 51), (16, 51), (19, 51), (14, 51), (24, 51), (20, 51),
(15, 51), (23, 51), (26, 51), (27, 51), (28, 51), (31, 51), (33, 51)}
{(43, 43), (36, 36), (39, 39), (37, 37), (50, 50)}

rank: 3
{(51, 51, 51)}
```