Conjugacy Class Enumeration in Transformation Semigroups (Conjclassts): User Manual

Part 6: Nilpotent and Idempotent Conjugacy Classes in Partial Transformation Semigroups

Clinton Oluranran Kayoh

August 12, 2025

Abstract

This document extends the previous manual on transformation semigroups to cover partial transformation semigroups, with special emphasis on nilpotent and idempotent elements and their conjugacy classes. The package now includes algorithms for generating various partial transformation semigroups and analyzing their algebraic properties.

Contents

1 Introduction to Partial Transformation Semigroups

1.1 Mathematical Background

A **partial transformation semigroup** $\mathcal{PT}_n$ on a finite set $X = \{1, 2, \ldots, n\}$ consists of all partial functions from X to X. Unlike full transformations, partial transformations may be undefined at some points, represented here by the symbol "-".

The size of $\mathcal{PT}_n$ is $(n+1)^n$, as each of the n points can map to any of the n elements or be undefined.

1.2 Nilpotent and Idempotent Elements

Definition 1. *An element $f \in \mathcal{PT}_n$ is **idempotent** if $f^2 = f \circ f = f$.*

Definition 2. *An element $f \in \mathcal{PT}_n$ is **nilpotent** if there exists $k > 0$ such that f^k is the zero transformation (maps everything to undefined).*

2 Core Functions for Partial Transformation Semigroups

2.1 Partial Transformation Semigroup Generation

2.1.1 PT_semigroup(n)

Description: Generates the partial transformation semigroup of degree n.

Mathematical Basis: Implements $\mathcal{PT}_n$, containing $(n+1)^n$ partial transformations.

Parameters: `n: int` - The degree of the partial transformation semigroup

Returns: `list` - List of tuples representing partial transformations

Complexity: $\mathcal{O}((n+1)^n)$ time and space complexity

```
def PT_semigroup(n):
elements = ["-", *range(1, n+1)]
combinations = itertools.product(elements, repeat=n)
combinations_list = list(combinations)
return combinations_list
```

Example 1. *For $n = 3$, the function returns all $4^3 = 64$ partial transformations.*

2.2 Order-Preserving Partial Transformations

2.2.1 check_P(t)

Description: Checks if a partial transformation preserves order.

Mathematical Basis: A partial transformation f is order-preserving if whenever $i \leq j$ and both $f(i), f(j)$ are defined, then $f(i) \leq f(j)$.

Parameters: `t: tuple` - A partial transformation

Returns: `bool` - True if the transformation is order-preserving

2.2.2 Order_Preserving_PT(tuples)

Description: Filters order-preserving partial transformations.

Parameters: `tuples: list` - List of partial transformations

Returns: `list` - Order-preserving partial transformations

2.3 Contraction Partial Transformations

2.3.1 contraction_check(t)

Description: Checks if a partial transformation is a contraction.

Mathematical Basis: For defined adjacent points i and $i+1$, $|f(i) - f(i+1)| \leq 1$.

Parameters: `t: tuple` - A partial transformation

Returns: `bool` - True if it's a contraction

```
                def contraction_check(t):
                if all(isinstance(x, str) for x in t):
                return True
                if not any(isinstance(x, int) for x in t):
                return False
                for i in range(len(t) - 1):
                if isinstance(t[i], int) and isinstance(t[i + 1], int):
                if abs(t[i] - t[i + 1]) > 1:
                return False
                return True
```

2.3.2 Contraction_PT(tuples)

Description: Generates contraction partial transformations.

Parameters: `tuples: list` - List of partial transformations

Returns: `list` - Contraction partial transformations

3 Nilpotent and Idempotent Analysis

3.1 Composition Function

3.1.1 compose(a, b, n)

Description: Computes the composition of partial transformations.

Mathematical Basis: Implements function composition $f \circ g$ for partial transformations.

Parameters: `a, b: tuple` - Partial transformations, `n: int` - Number of compositions

Returns: `tuple` - Result of composition

```
                def compose(a, b, n):
                result = a
                for _ in range(n):
                result = tuple(result[i - 1] if isinstance(i, int) else i for i in b)
                return result
```

3.2 Nilpotent and Idempotent Detection

3.2.1 compute_nilpotent_idempotent(elements)

Description: Identifies nilpotent and idempotent elements in a set.

Mathematical Basis: Tests the algebraic properties $f^2 = f$ (idempotent) and $f^k = 0$ (nilpotent).

Parameters: `elements: list` - List of partial transformations

Returns: `tuple` - Two lists: nilpotent elements and idempotent elements

```
def compute_nilpotent_idempotent(elements):
nilpotent = []
idempotent = []
for element in elements:
is_nilpotent = True
is_idempotent = True
# Check for nilpotency
if compose(element, element, 10) != tuple(["-"] * len(element)):
is_nilpotent = False
# Check for idempotence
if compose(element, element, 1) != element:
is_idempotent = False
if is_nilpotent:
nilpotent.append(element)
if is_idempotent:
idempotent.append(element)
return nilpotent, idempotent
```

4 Enhanced Conjugacy Class Analysis

4.1 Extended conjugacy_class function

Description: Enhanced version that identifies nilpotent and idempotent conjugacy classes.

Features:
- Partitions transformations into conjugacy classes via graph isomorphism
- Identifies nilpotent elements within each class
- Identifies idempotent elements within each class
- Provides summary statistics

```
def conjugacy_class(tuples_list):
classes = {}
# Graph isomorphism classification
for i, tpl in enumerate(tuples_list, start=1):
G = nx.DiGraph()
for j, el in enumerate(tpl, start=1):
G.add_edge(j, el)
found = False
for k, v in classes.items():
if nx.is_isomorphic(G, v[0]):
v.append(tpl)
found = True
break
if not found:
classes[len(classes)+1] = [G, tpl]

# Nilpotent and idempotent analysis
nilpotent_classes = 0
idempotent_classes = 0
for k, v in classes.items():
nilpotent, idempotent = compute_nilpotent_idempotent(v[1:])
if nilpotent:
nilpotent_classes += 1
if idempotent:
idempotent_classes += 1

return classes, nilpotent_classes, idempotent_classes
```

5 Mathematical Properties and Theorems

Theorem 1. *In $\mathcal{PT}_n$, the conjugacy class of a partial transformation is determined by the isomorphism type of its functional graph, where undefined points correspond to isolated vertices with no outgoing edges.*

Theorem 2. *A partial transformation is idempotent if and only if for every defined point x, $f(x)$ is a fixed point of f.*

Theorem 3. *A partial transformation is nilpotent if and only if its functional graph contains no cycles and all paths eventually lead to undefined states.*

Remark 1. *The zero transformation (all points undefined) is both nilpotent and idempotent, serving as the zero element in $\mathcal{PT}_n$.*

6 Usage Examples

6.1 Generating Specialized Semigroups

```
# Generate order-preserving contraction partial transformations
n = 3
PT_n = PT_semigroup(n)
contraction_PT = Contraction_PT(PT_n)
order_preserving_contraction = Order_Preserving_PT(contraction_PT)

print(f"Order-preserving contraction transformations: {len(
    order_preserving_contraction)}")
```

6.2 Analyzing Algebraic Properties

```
# Analyze nilpotent and idempotent conjugacy classes
classes, nilpotent_count, idempotent_count = conjugacy_class(
    order_preserving_contraction)

print(f"Total conjugacy classes: {len(classes)}")
print(f"Nilpotent conjugacy classes: {nilpotent_count}")
print(f"Idempotent conjugacy classes: {idempotent_count}")

# Display specific class information
for class_id, elements in classes.items():
nilpotent, idempotent = compute_nilpotent_idempotent(elements[1:])
if nilpotent:
print(f"Class {class_id} contains {len(nilpotent)} nilpotent elements")
if idempotent:
print(f"Class {class_id} contains {len(idempotent)} idempotent elements")
```

7 Advanced Applications

7.1 Structure Theory

The package enables investigation of:

- Green's relations in partial transformation semigroups
- Lattice of idempotents
- Nilpotent ideals and radicals
- Representation theory of partial transformation semigroups

7.2 Combinatorial Applications

- Enumeration of special classes of partial transformations
- Asymptotic behavior of conjugacy class numbers
- Connections with Catalan numbers and other combinatorial sequences

8 Limitations and Future Development

8.1 Current Limitations

- Exponential complexity limits analysis to small n ($n \leq 5$)
- Basic nilpotency test (fixed 10 iterations)
- No optimization for large semigroups

8.2 Planned Enhancements

TODO:

- Efficient nilpotency detection using graph analysis
- Algebraic structure analysis (Green's relations)
- Integration with semigroup theory databases
- Asymptotic analysis tools

References

[1] Howie, J. M. (1995). *Fundamentals of Semigroup Theory*. Oxford University Press.

[2] Ganyushkin, O., Mazorchuk, V. (2008). *Classical Finite Transformation Semigroups*. Springer.

[3] Arbib, M. A. (1968). *Algebraic Theory of Machines, Languages, and Semigroups*. Academic Press.

[4] Abusarris, H., & Ayık, G. (2023). On the rank of generalized order-preserving transformation semigroups. *Turkish Journal of Mathematics*. https://doi.org/10.55730/1300-0098.3420.

[5] Adan-Bante, E. (2005). On nilpotent groups and conjugacy classes. *arXiv: Group Theory*, 345-356.

[6] Ahmad, A., Magidin, A., & Morse, R. (2012). Two generator p-groups of nilpotency class 2 and their conjugacy classes. *Publicationes Mathematicae Debrecen*, 81, 145-166. https://doi.org/10.5486/PMD.2012.5114.

[7] Akinwunmi, S.A. & Makanjuola, S.O. (2019). Enumeration Partial Contraction Transformation Semi-groups. *Journal of the Nigerian Association of Mathematical Physics*, 29(B), 70-77.

[8] Akinwunmi, S.A., Mogbonju, M.M., Adeniji, A.O., Oyewola, D.O., Yakubu, G., Ibrahim, G.R., & Fatai, M.O. (2021). Nildempotency Structure of Partial One-One Contraction CIn Transformation Semigroups. *International Journal of Research and Scientific Innovation (IJRSI)*, 8(1), 230-236.

[9] Mogbonju, M.M., Gwary, T.M., & Ojeniyi, A.B. (2014). Presentation of Conjugacy Classes in the Partial Order-Preserving Transformation Semigroup with the aid of graph. *Mathematical Theory and Modeling*, 4, 110-142.

[10] Mohammadian, A., & Erfanian, A. (2018). On the nilpotent conjugacy class graph of groups. *Journal of Algebra and Its Applications*, 37, 77-90. https://doi.org/10.1142/S0219498818500407.

[11] Ugbene, I.J., Eze, E.O., & Makanjuola, S.O. (2013). On the Number of Conjugacy Classes in the Injective Order-Decreasing Transformation Semigroup. *Pacific Journal of Science and Technology*, 14(1), 182-186.

[12] Ugbene, I.J. & Makanjuola, S.O. (2012). On the Number of Conjugacy Classes in the Injective Order-preserving Transformation Semigroup. *Icastor Journal of Mathematical Sciences*, 6(1), 1-8.

[13] Ugbene, I.J., Suraju, O., & Ugochukwu, N. (2021). Digraph of the full transformation semigroup. *Journal of Discrete Mathematical Sciences and Cryptography*, 25(8), 2457-2465. https://doi.org/10.1080/09720529.2020.1862955.