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

Part 4: Nilpotent and Idempotent Conjugacy Classes in Full Transformation Semigroups

Clinton Oluranran Kayoh

August 08, 2025

Abstract

This document extends the user manual for the Python package dedicated to enumerating conjugacy classes in transformation semigroups. This continuation focuses specifically on **nilpotent and idempotent elements** within full transformation semigroups, which represent fundamental algebraic structures with important combinatorial and theoretical properties. The package implements specialized algorithms for identifying and classifying these special elements within conjugacy classes.

Contents

1 Introduction to Nilpotent and Idempotent Elements

1.1 Mathematical Background

In semigroup theory, nilpotent and idempotent elements play crucial roles in understanding the algebraic structure:

Definition 1. *An element $f \in \mathcal{T}_n$ is **idempotent** if $f \circ f = f$ or if it path structure contains circuit path. Idempotents represent "projections" or "fixed points" in the semigroup.*

Definition 2. *An element $f \in \mathcal{T}_n$ is **nilpotent** if there exists some positive integer k such that f^k is a constant transformation or if it path structure contains proper path. Nilpotents represent transformations that eventually "collapse" the entire set to a single point.*

1.2 Algebraic Significance

- Idempotents form a natural partial order within the semigroup
- Nilpotents represent the "zero-like" elements in the structure
- The distribution of these elements across conjugacy classes reveals deep algebraic properties
- They play key roles in Green's relations and the structure theory of semigroups

2 Core Functions for Nilpotent and Idempotent Analysis

2.1 Composition Operations

2.1.1 compose(a, b)

Description: Computes the composition of two transformations $a \circ b$.

Mathematical Basis: Implements semigroup composition: $(a \circ b)(i) = a(b(i))$.

Parameters: `a: tuple, b: tuple` - Two transformations to compose

Returns: `tuple` - The composed transformation

```
def compose(a, b):
result = []
for i in range(len(a)):
result.append(a[b[i] - 1])
return tuple(result)
```

2.1.2 compose_n(a, b, n)

Description: Computes the n-fold composition $a \circ b \circ \cdots \circ b$ (n times).

Mathematical Basis: Implements iterative composition for checking nilpotency.

Parameters: `a: tuple, b: tuple, n: int` - Transformations and iteration count

Returns: `tuple` - The n-fold composed transformation

```
def compose_n(a, b, n):
result = a
for _ in range(n-1):
temp = []
for i in range(len(a)):
temp.append(a[b[i] - 1])
result = tuple(temp)
return result
```

2.2 Nilpotent and Idempotent Detection

2.2.1 compute_nilpotent_idempotent(elements)

Description: Identifies nilpotent and idempotent elements within a set of transformations.

Mathematical Basis:
- Idempotency: $f^2 = f$
- Nilpotency: f^k is constant for some k (approximated by checking if f^{10} equals identity)

Parameters: `elements: list` - List of transformations to analyze

Returns: `tuple` - Two lists: (nilpotent_elements, idempotent_elements)

```
                def compute_nilpotent_idempotent(elements):
                nilpotent = []
                idempotent = []

                for element in elements:
                is_nilpotent = True
                is_idempotent = True

                # Check for nilpotency
                if compose_n(element, element, 10) != tuple(range(1, len(elements[0]) + 1)):
                is_nilpotent = False

                # Check for idempotence
                if compose(element, element) != element:
                is_idempotent = False

                # Append to respective lists
                if is_nilpotent:
                nilpotent.append(element)
                if is_idempotent:
                idempotent.append(element)

                return nilpotent, idempotent
```

2.3 Enhanced Conjugacy Class Analysis

2.3.1 conjugacy_class(tuples_list)

Description: Extended version that identifies nilpotent and idempotent conjugacy classes within the full classification.

Mathematical Basis: A conjugacy class is nilpotent (resp. idempotent) if all its elements are nilpotent (resp. idempotent).

Parameters: `tuples_list: list` - List of transformations to classify

Returns: `dict` - Enhanced classification with special element identification

3 Mathematical Properties and Characterization

3.1 Characterization of Idempotents

Theorem 1. *A transformation $f \in \mathcal{T}_n$ is idempotent if and only if:*

1. *$f(x) = x$ for all x in the image of f*
2. *The restriction of f to its image is the identity function*

Proof. If f is idempotent, then $f(f(x)) = f(x)$ for all x, so $f(x)$ is a fixed point. Conversely, if f acts as identity on its image, then $f(f(x)) = f(x)$. □

3.2 Characterization of Nilpotents

Theorem 2. *A transformation $f \in \mathcal{T}_n$ is nilpotent if and only if:*

1. *There exists a unique sink component in the functional graph*
2. *All elements eventually map to this sink under iteration*
3. *The sink is a single vertex (constant transformation)*

3.3 Conjugacy Class Structure

Theorem 3. *Within a conjugacy class in $\mathcal{T}_n$:*

- *Either all elements are idempotent, or none are*
- *Either all elements are nilpotent, or none are*
- *A class can be both nilpotent and idempotent only if it consists of constant maps*

4 Usage Examples

4.1 Basic Nilpotent and Idempotent Analysis

```
# Generate full transformation semigroup
T3 = FT_semigroup(3)

# Analyze nilpotent and idempotent elements
nilpotent, idempotent = compute_nilpotent_idempotent(T3)
print(f"Total nilpotent elements: {len(nilpotent)}")
print(f"Total idempotent elements: {len(idempotent)}")

# Display examples
print("Example nilpotents:", nilpotent[:3])
print("Example idempotents:", idempotent[:3])
```

4.2 Enhanced Conjugacy Class Analysis

```
# Perform complete conjugacy class analysis with special elements
classes_analysis = conjugacy_class(T3)

# The function automatically prints:
# - Total number of conjugacy classes
# - Classification of each class
# - Identification of nilpotent and idempotent classes
# - Counts of special conjugacy classes
```

4.3 Composition Verification

```
# Verify composition operations
f = (1, 1, 2)
g = (2, 3, 1)
composition = compose(f, g)
print(f"f * g = {composition}")

# Check idempotency
is_idempotent = compose(f, f) == f
print(f"Is f idempotent? {is_idempotent}")

# Check nilpotency (approximate)
is_nilpotent = compose_n(f, f, 10) == tuple(range(1, len(f)+1))
print(f"Is f nilpotent? {is_nilpotent}")
```

5 Theoretical Results and Observations

5.1 Distribution in $\mathcal{T}_3$

Based on the computational results for $n = 3$:

Property	Count	Percentage	Notes
Total Elements	27	100%	
Conjugacy Classes	7	25.9%	
Idempotent Classes	3	42.9%	Classes 1, 3, 4
Nilpotent Classes	2	28.6%	Classes 4, 5
Idempotent Elements	10	37.0%	
Nilpotent Elements	4	14.8%	

Table 1: Distribution of special elements in $\mathcal{T}_3$

5.2 Structural Patterns

Theorem 4. *In $\mathcal{T}_n$, the constant maps form a single conjugacy class that is both nilpotent and idempotent.*

Theorem 5. *The number of idempotents in $\mathcal{T}_n$ is given by:*

$$\sum_{k=1}^{n} \binom{n}{k} k^{n-k}$$

where k is the size of the image.

6 Advanced Mathematical Insights

6.1 Green's Relations and Special Elements

Definition 3. *The $\mathcal{D}$-class of an idempotent is a maximal subgroup of the semigroup. Idempotents in the same $\mathcal{D}$-class are conjugate.*

Theorem 6. *For nilpotent elements, the $\mathcal{J}$-relation captures the "depth" of nilpotency - how quickly the transformation collapses to a constant map.*

6.2 Combinatorial Interpretation

The functional graph perspective provides clear interpretations:

- **Idempotents**: Each component is a star with a fixed point at the center
- **Nilpotents**: The graph has a unique sink that attracts all vertices
- **Conjugacy**: Graph isomorphism preserves these structural properties

7 Limitations and Mathematical Refinements

7.1 Current Limitations

- Nilpotency check uses fixed iteration depth (10) - may miss some nilpotents
- No distinction between different nilpotency indices
- No characterization of $\mathcal{D}$-classes or Green's relations
- Memory-intensive for larger n

7.2 Mathematical Refinements

TODO:

- Implement precise nilpotency index computation
- Add Green's relations classification
- Incorporate structure theory of regular $\mathcal{D}$-classes
- Optimize using cycle-chain decomposition

8 Computational Complexity

8.1 Time Complexity

- Composition: $\mathcal{O}(n)$ per operation
- Nilpotency check: $\mathcal{O}(n^2)$ worst case (fixed iterations)
- Idempotency check: $\mathcal{O}(n)$
- Full analysis: $\mathcal{O}(n^n \cdot n^2)$ dominated by graph isomorphism

8.2 Space Complexity

- Storage of transformations: $\mathcal{O}(n^n)$
- Graph representations: $\mathcal{O}(n^2)$ per transformation
- Conjugacy classes: $\mathcal{O}(n^n)$ total

9 Conclusion

This extension provides sophisticated tools for analyzing the fundamental algebraic structures within transformation semigroups. The identification and classification of nilpotent and idempotent elements within conjugacy classes reveals deep connections between combinatorial structure and algebraic properties.

The implementation bridges the gap between abstract semigroup theory and computational experimentation, enabling researchers to verify theoretical predictions and discover new patterns in finite semigroups.

References

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

[2] Lipson, J. D. (1972). *Idempotents in full transformation semigroups*. Semigroup Forum.

[3] Gomes, G. M. S., & Howie, J. M. (2005). *On the ranks of certain finite semigroups of transformations*. Mathematical Proceedings.

[4] Cameron, P. J. (1999). *Permutation Groups*. Cambridge University Press.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.