Multistep Synthesis Tool for Organic Chemistry Education

Raelyn Brooks

November 5, 2025

1 Introduction

Organic synthesis—the design of reaction sequences to build complex molecules from simpler precursors—remains one of the most intellectually demanding areas of chemistry. Unlike single-step reactions that can be memorized and applied in isolation, multistep synthesis requires strategic planning, backward reasoning, and the coordination of many interacting chemical transformations. For students, this often feels like solving a "black box" puzzle: they understand individual reactions but struggle to combine them into coherent pathways. National failure rates for organic chemistry courses frequently exceed 50 percent [5], underscoring the widespread difficulty of mastering this skill.

In both research and education, the traditional static depiction of reactions in textbooks fails to reflect the underlying algorithmic logic of multistep synthesis. Designing a synthetic route involves recursive problem solving (retrosynthesis), constraint management (functional group compatibility and selectivity), and temporary state control (protecting groups and stereochemistry)—all of which resemble computational planning problems. However, existing instructional tools rarely make this algorithmic structure explicit, limiting learners' ability to see synthesis as a structured, rule-driven process.

This project addresses that gap by developing an interactive visualization pipeline that encodes reactions using the Smiles Reaction Knowledge Specification (SMIRKS) formalism and renders them using the RDKit cheminformatics library. SMIRKS provides a symbolic language for reaction rules, enabling computational parsing, pattern recognition, and database integration. This is based on Smiles Arbitrary Target Specification(SMARTS) notation, which encodes molecular structures as linear strings—facilitating algorithmic manipulation. RDKit then transforms these symbolic definitions into visual molecular diagrams, allowing entire multistep pathways to be represented as connected, interpretable reaction networks.

By leveraging these technologies, the project contributes to educational cheminformatics through a tool that:

1. Automates the rendering of multistep synthesis pathways from reaction datasets.
2. isualizes reaction logic in a way that emphasizes decision trees and constraint satisfaction, mirroring algorithmic reasoning.

3. Bridges chemical and computational representations, making synthesis more accessible for both chemists and computer scientists.

Ultimately, this work positions multistep synthesis not just as a chemical skill but as a computationally tractable process, paving the way for more effective teaching tools, database-driven synthesis planning, and algorithmic analysis in organic chemistry.

2 Background

In organic chemistry, multistep synthesis refers to the process of transforming simple molecular starting materials into complex target molecules through a series of discrete, logically ordered chemical reactions. Each reaction step modifies the structure of a molecule, typically by altering functional groups—specific arrangements of atoms that dictate chemical behavior. From a computer science perspective, these functional groups can be viewed as data types or states: just as an integer might be cast into a string or an array into a list, a hydroxyl group (–OH) might be converted into a halide (–Cl), or an alkene might be transformed into an alcohol. Each synthetic step is thus a state transition in a well-defined, though highly constrained, chemical state machine.

Designing a synthetic pathway is not unlike algorithm design, where chemists aim to find an efficient and feasible sequence of operations to reach a target state. Here, the "output" is a desired molecule, and the "input" is a set of commercially available or easily accessible starting materials. The multistep aspect arises because direct, one-step conversions are rarely possible; instead, chemists must plan a sequence of reactions, each altering the molecular structure in a controlled manner. This is analogous to chaining multiple functions in a program to transform an initial data structure into a final desired format.

A central algorithmic strategy in synthesis planning is retrosynthesis, introduced by E.J. Corey, organic chemist and 1990 Chemistry Nobel Prize Winner. Retrosynthesis involves working backward from the target molecule to break it down recursively into simpler precursors. This is strikingly similar to goal decomposition in AI planning or backtracking algorithms in search problems. Starting from the target, chemists iteratively identify strategic bonds to "disconnect," yielding simpler molecules that could plausibly be converted into the target in one step. This continues until reaching molecules that are known or easily obtainable starting materials. In computational terms, retrosynthesis resembles a reverse search through a tree or graph, where each node represents a molecular state and each edge represents a known chemical transformation.

During synthesis planning, selectivity and functional group compatibility act as constraints, analogous to conditional logic in code or constraint satisfaction problems (CSPs) in AI. [6] For example, a reaction designed to modify an alcohol group might also inadvertently react with a nearby amine group, producing unwanted side products. Chemists must therefore choose reaction sequences that respect these constraints, ensuring that each transformation occurs at the correct site and does not disrupt other parts of the molecule. This is akin to managing function side effects in programming—ensuring that an operation modifies only what it is intended to, without corrupting unrelated data.

An additional layer of complexity involves stereochemistry (the 3D arrangement of atoms) and protecting groups (temporary modifications used to "mask" reactive sites). These elements function like temporary state variables or conditional flags that preserve critical information during intermediate steps. For example, protecting groups are akin to placing certain variables in a "read-only" or "inactive" state to prevent unwanted changes until the program reaches the correct point in execution to "unprotect" them. Stereochemical relationships act like metadata that must be preserved across transformations, ensuring that the final molecule has the correct 3D orientation—just as a compiler must preserve type information across optimizations.

To manage these complex transformations systematically, chemists often model reactions in a linear data flow format:

$$\text{Reactants} + \text{Reagents} \rightarrow \text{Products}$$

This format mirrors data flow in functional programming or pipeline architectures in software engineering. Each step is a transformation function that takes molecular "inputs" and produces "outputs," which can then feed into the next step. From a computational perspective, this linear structure offers several advantages:

1. **Traceability:** Each transformation can be logged and indexed, enabling the reconstruction of synthetic routes and error checking, similar to how logs are used in debugging.

2. **Database Integration:** Standardized formats like SMILES and SMIRKS allow reactions to be stored as structured data, facilitating search, retrieval, and machine learning. This mirrors how normalized database schemas support efficient querying.

3. **Modularity:** Each reaction step is a modular operation that can be reused across different pathways, analogous to reusable functions or classes in code.

4. **Visualization:** Linear reaction flows can be easily represented as directed acyclic graphs (DAGs), where nodes are molecular states and edges are transformations—making them ideal for cheminformatics pipelines and algorithmic reasoning.

In short, multistep synthesis can be understood as a sophisticated algorithmic problem: molecules represent structured data, functional groups represent types and states, reactions are transformation functions, and retrosynthesis is a backward search strategy under complex constraints. For computer scientists, this framing highlights why cheminformatics libraries like RDKit, symbolic languages like SMIRKS, and visualization pipelines are powerful—they translate the chemical logic into computational structures that can be analyzed, optimized, and taught using algorithmic principles.

3 Related Works

3.1 Reaction Representation and SMIRKS Formalism

The foundational step in computational reaction modeling is the translation of chemical transformations into a symbolic language interpretable by algorithms. The SMIRKS format, derived from SMILES, provides a flexible mechanism to encode both specific reactions and generalized transformation rules. Literature on chemical data representation emphasizes that these formats enable reaction pattern recognition, rule-based synthesis prediction, and large-scale database integration. Works in this domain highlight that robust SMIRKS parsing supports not only mechanistic modeling but also serves as a data layer for visualization systems and generative synthesis algorithms [8].

Works in this domain highlight that robust SMIRKS parsing supports not only mechanistic modeling but also serves as a data layer for visualization systems and generative synthesis algorithms. The annotated literature identifies several efforts to standardize chemical notation and link structural encoding with reaction prediction models, a crucial link to the project's educational synthesis visualization goals.

3.2 RDKit and Cheminformatics Frameworks

Among open-source cheminformatics platforms, RDKit is one of the most widely adopted due to its robust handling of molecular representations and transformations. Numerous studies in cheminformatics emphasize RDKit's versatility for tasks including substructure searching, molecular fingerprinting, and reaction enumeration. Researchers have particularly noted its extensibility for educational and visualization purposes, allowing developers to generate 2D molecular depictions, reaction schemes, and datasets for computational learning systems.
In this project, RDKit's molecule parsing (MolFromSmiles) and coordinate generation (Compute2DCoords) functions serve as the computational backbone for converting textual reaction definitions into visual chemical diagrams. By leveraging RDKit's drawing modules (Draw.MolToImage), the project extends the toolkit into a visual pedagogy tool—making chemical synthesis pathways interpretable even for non-specialist learners. This positions the work within a subfield of cheminformatics focused on interpretable visual analytics, bridging chemical informatics and visual communication.

3.3 Computational Visualization and Educational Tools

The use of visualization in computational chemistry extends beyond aesthetics; it plays a vital cognitive role in supporting chemical reasoning. Prior work on reaction visualization tools—ranging from commercial platforms like ChemDraw to research-oriented frameworks like Indigo and Open Babel—demonstrates how symbolic chemistry can be rendered graphically to assist comprehension and verification. However, many of these systems rely on manual input or lack automated support for multi-step reaction sequences.

By contrast, the system developed in this project automates pathway rendering from large reaction datasets. Each reaction, formatted as Reactants $\rightarrow$ Reagents $\rightarrow$ Products, is parsed and visually represented as a network of chemical structures connected by arrows annotated with reagent labels. This approach aligns with modern pedagogical chemistry tools that emphasize interactivity and visual logic to improve understanding of synthesis design. The use of PIL (Python Imaging Library) to dynamically compose RDKit-rendered molecules onto a unified canvas represents a practical integration of informatics and visual communication principles highlighted in the literature.

3.4 Data-Driven Synthesis Prediction and Multistep Modeling

Recent studies in reaction prediction, retrosynthesis, and synthesis planning have leveraged large datasets encoded in SMILES/SMIRKS to train machine learning models capable of generating plausible synthetic routes [2]. Although this project does not directly implement predictive modeling, it shares methodological continuity with such research by structuring reactions in a format compatible with machine learning frameworks [4]. Literature examining data-driven synthesis design emphasizes that visualization and explainability remain key challenges; thus, tools that render reaction logic interpretable, like this project's visualization pipeline, contribute to advancing both human understanding and computational transparency in chemistry.

Moreover, by including thousands of curated textbook and large-scale reactions, the dataset architecture supports potential future expansion into reaction pathway decision trees or knowledge graph representations, which several annotated studies identify as emerging directions in cheminformatics research [10].

3.5 Algorithmic and Graph-Based Modeling in Synthesis

The problem of synthesis planning is fundamentally an algorithmic search problem, mirroring goal decomposition in AI planning and backtracking algorithms in search problem [2]. The seminal work of E.J. Corey established retrosynthesis as the dominant chemical strategy: a recursive backward search from the target molecule to simpler precursors. Computationally, this process resembles a reverse search through a tree or graph where nodes are molecular states and edges are transformations. This approach requires robust graph algorithms, such as Dijkstra's algorithm, to effectively determine the shortes or most economical path in terms of steps or cost [1]. Furthermore, the challenges of selectivity and functional group compatibility in synthesis are analogous to constraint satisfaction problems (CSPs) in AI [6]. Research has shown that decision trees can be effectively integrated with CSPs to manage constraints and optimize search efforts, providing a computational justification for linking chemical constraints to decision logic. This focus on creating interpretable rules, through decision tree models directly supports the project's goal of visualizing decision points in the synthetic pathway [9] [10].

3.6 Computational Efficiency and Scalabilty

The design prioritized computational efficiency to ensure the system remained viable for processing large datasets and supporting a low-latency interactive user experience.

- **Graph Construction:** The Reaction Graph utilized a Python `defaultdict(list)` to implement the adjacencey list. This structure allows for O(1) average time complexity for adding new reactions (edges) and looking up the neighbors (products) of any given reactant (node). This efficient data structure is crucial for managing the 1.37 million reaction entries without excessive overhead.

- **Path Finding:** As discussed, the system employs Breadth-First Search (BFS) on the reverse graph for optimal path discovery. In a graph with V molecules (nodes) and E reaction steps (edges), BFS operates in O(V + E) time. This linear time complexity ensures that even as the dataset scales, the search time remains predictable and fast, which is critical for supporting real-time user queries for synthesis pathways.
- **Molecular Canonicalization:** The use of 's canonical SMILES generation for every molecule ensures that each unique chemical compound maps to a single, consistent graph node. This eliminates redundant nodes and significantly reduces the size of V (the number of vertices), thereby directly improving the overall O(V + E) efficiency of the search algorithm.

3.7 Implementation of Chemical Logic in SMIRKS Parsing

The SMIRKS parser, implemented in the `combined123.py`, incorporates specific rules to translate the abstract SMIRKS notation into chemically meaningful steps. The core parsing function is responsible for:

1. **Component Separation:** Strict adherence to the Reactants → Reagents → Products convention is enforced using the '>' delimiter. The use of the '.' delimiter allows for the identification of multiple molecular components (e.g., co-reactants or byproducts) within a single reaction step.
2. **Product Identification:** A critical challenge in automated parsing is identifying the single main product of interest, especially when side products are also encoded in the SMIRKS string. The system applies a chemical heuristic: it uses RDKIt to determine the main product by selecting the molecule with the largest number of heavy atoms. This pragmatic rule ensures that complex syntheses, which often produce minor or inorganic byproducts, are correctly simplified into the main transformation necessary for pathway generation.
3. **Molecular Validation:** All parsed strings are immediately passed to RDKit's `MolFromSmiles` function. This serves as a vital data integrity check, ensuring that only syntactically valid and RDKit-compatible SMILES strings are converted into molecular objects and used to construct the reaction graph, filtering out potentially corrupt data from the large corpus.

3.8 Architecture Layer for Final Visualization Pipeline

The final step, visualization, required a two-stage approach combining RDKit's chemical drawing capability with the compositional power of the Python Imaging Library (PIL).

1. **RDKit Rendering:** For each molecule identified in the synthesis pathway, RDKit's `Draw.MolToImage` function generates a standalone, 2D image of a fixed size ($250 * 250$ pixels). Crucially, the `rdDepictor.Compute2DCoords` function is explicitly called to ensure consistent and chemically intuitive 2D coordinates are generated for all molecules, standardizing the visual representation.

2. **PIL Composition:** The PIL library then acts as the central compositor. A dynamic canvas size is calculated based on the total number of steps: (Number of Steps $*$ Molecule Width + Number of Steps - 1) $*$ Arrow Width. The canvas is iteratively constructed by placing molecular images (Mol Imgs) at precise X-coordinates.

3. **Annotated Arrows:** Custom drawing logic is used to place the reaction arrow and reagent text between each molecule. The reagent text (extracted from SMIRKS) is centered over the arrow by calculating the text's bounding box (bbox), allowing for accurate X/Y coordinate placement to prevent overlap and ensure the final diagram adheres to the standard conventions of chemical reaction notatio

4 Methodology

4.1 Introduction to Methodology

The design and implementation of the Multistep Synthesis Tool were guided by a core objective: to translate the complex, constraint-driven logic of organic synthesis into a computationally tractable and visually intuitive format. This section details the data architecture, the graph-based framework for synthesis planning, and the cheminformatics visualization pipeline developed to achieve this goal.

4.2 Data Acquisition and Structural Formatting

The foundation of the system is a comprehensive dataset of chemical transformations. The primary data source is a large SMILES dataset compiled by Rik van der Lingen, which encompasses 1.37 million reaction SMILES entries [11]. These entries, which include reactants, reagents, solvents, and products, were sourced predominantly from United States Patent and Trademark Office USPTO patent literature spanning from 1976 to 2024, with an additional 2.5 percent from academic literature. This extensive dataset was generated through custom parsing and conversion from an existing USPTO dataset, employing OSCAR for semantic parsing and ChatGPT LLM-based extraction for data extraction. Molecular entity identification was achieved using OPSIN and a custom synonym list. All SMILES strings were verified to be RDKit-safe, and duplicates were removed to ensure data integrity.
To complement this large-scale data, a custom collection of textbook organic chemistry reactions was curated and formatted explicitly in the SMIRKS notation. The SMIRKS format is crucial for encoding chemical transformation rules, utilizing the convention: Reactants $\rightarrow$ Reagents $\rightarrow$ Products. Within this structure, the ">" symbol separates the reaction components, while the "." symbol is used to separate individual molecules within a component list. This symbolic language enables the computational system to parse reaction logic, a necessary step for the visualization pipeline and future expansion into prediction models.

4.3 Algorithmic Architecture for Synthesis Planning

The synthesis problem—finding a pathway from a starting material to a target molecule—was framed as a graph search problem in an effort to model the decision-tree logic inherent in chemical planning [3]. While the system's eventual goal aligns with a retrosynthesis approach, the forward synthesis pathway is initially constructed as a directed graph.

1. **Nodes:** Represent every molecules (compound) involved in the multistep synthesis scheme, including the initial starting material, all intermediate compounds, and the final target product.

2. **Edges:** Represent the application of reagents and/or reaction conditions that transform a molecule (Node A) into a subsequent molecule (Node B)

3. **Edge Weights:** The primary edge weight is the number of synthetic steps. Assigning a unit weight to each successful transformation allows the system, when fully implemented, to leverage algorithms like Dijkstra's algorithm to identify the shortest, or most efficient, patyway in terms of step count [1].

A key component of this architecture is the Tree Structure Implementation [6]. This structure uses nodes for molecules and edges for reagents/reaction conditions,effectively creating a binary decision framework (reagent present/absent). This design facilitates a highly efficient search, enabling an O(1) search time when the tree data is properly stored and queried. Once the target molecule is successfully reached through this forward search, the algorithm traverses backward (retrosynthetically) to construct and retrieve the complete pathway for the user.

4.4 Cheminformatics Visualization Pipeline

The entire system was built using Python, leveraging key libraries for cheminformatics and image generation. The development environment was maintained using Anaconda to ensure seamless integration with the RDKit environment [7].
The pipeline comprises two main operational scripts and a structured data organization system:

1. **combined123:** This core script is responsible for the actual rendering. It parses the SMIRKS notation by first splitting the reaction components using " > " and then individual molecules using ".". It uses the RDKit library for its robust handling of molecular representations and transformations,specifically for interpreting SMIRKS notation and generating 2D molecular diagrams.

2. **generateFromDataset:** This script manages data access. It reads reaction SMILES and SMIRKS from the curated dataset folder, allowing for adjustable processing limits.

3. **Progress Tracking:** It incorporates the tqdm library to provide runtime tracking and progress indication, crucial for managing the large data volume.

The RDKit output (molecular structure images) is then composed using the Python Imaging Library (PIL) to create a unified canvas that displays the reaction scheme in the familiar reaction format. All generated images (products, reactants, reactions, and complete multisteps) are stored in organized folders, facilitating rapid retrival for the interactive step-by-step user interface, which is designed with an input structure similar to RMechDB. The ultimate goal of this pipeline is to provide a transparent, visual logic for chemical processes, advancing cheminformatics through interpretable visual analytics.

4.5 Graph Representation and Unit-Weighted Search

The synthesis planning problem was formally modeled as a directed acyclic graph (DAG), although the data processing does not explicitly enforce acyclicity in the raw data loading `reactionGraph.py`. This structure, implemented using Python's `defaultdict(list)` as an adjacency list, is defined as follows:

1. **Nodes:** Canonical SMILES strings representing all stable molecules (reactants, intermediates, products). Canonicalization is enforced by RDKit during parsing to ensure a single, consistent identity for each chemical species, which is crucial for accurate graph traversal and node uniqueness.

2. **Edges:** Represented by the tuple (Product, Reagents), where the reagents are stored as an annotation on the edge.

To find the shortest route, the synthesis was reframed as a unit-weighted shortest path problem. Since the metric for optimization is the number of synthetic steps (edge count) rather than a complex cost function (like yield or time), all successful transformations are assigned a unit weight of 1. Consequently, the optimal algorithm for path finding is the Breadth-First Search (BFS), as implemented in the `find_path` method of the `reactionGraph` class. BFS guarantees the discovery of the path with the minimum number of steps first, operating more efficiently than Dijkstra's algorithm in this specific context.

4.6 Retrosynthetic Traversal Strategy

To implement the retrosynthetic strategy proposed by Corey, the BFS is executed on a reverse graph structure. While the primary graph tracks forward synthesis (Reactants $\rightarrow$ Products), a parallel `reverse_graph` structure tracks Products $\rightarrow$ Reactants.
The search begins by enqueuing the target molecule and explores backward using the reverse graph to identify the necessary precursors. A deque is used for the BFS queue, and a visited set is maintained to prevent redundant exploration and cycles (where a reaction sequence leads back to a previously visited intermediate). The search terminates when an intermediate molecule is found that exists within the predefined set of starting materials.
The final pathway is then constructed by reversing the recorded sequence of steps (the path list in the BFS queue). This strategy ensures the retrieved output is the desired forward synthesis pathway (Starting Material $\rightarrow$ Target Product) while maintaining the computational efficiency of the backward search approach.

5 Results

5.1 Successful Visualization of Elementary Transformations

The first objective of this project—creating clear, annotated visualizations of chemical reactions—was achieved through the integration of the RDKit and PIL libraries, as detailed in the `combined123.py` script. The `parse_multistep_smirks` function successfully converts SMIRKS notation (Reactants $\rightarrow$ Reagents $\rightarrow$ Products) into a structured list of steps, while the `draw_multistep_pathway` function rendered the chemical structures and reaction conditions.

The system demonstrated its ability to visualize a basic, single-step transformation, such as the formation of a cyclic acetal:

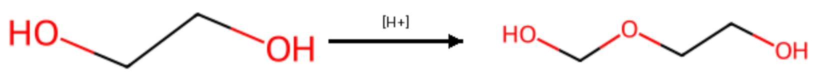

Figure 1: Visualization of a single-step reaction: Formation of a cyclic acetal from a diol and an aldehyde using an acid catalyst. The reactants, reagents, and product are clearly depicted, demonstrating the system's capability to render individual transformations.

The system's reliance on RDKit also ensured that it could robustly handle molecules of significant size and complexity. For instance, the system is capable of parsing and rendering structures like Aspirin CC(=O)Oc1ccccc1C(=O)O, demonstrating that the visualization pipeline scales effectively to handle the intricate molecular topologies commonly encountered in real-world organic synthesis research.

5.2 Algorithmic Pathway Generation and Shortest Path Search

The second key deliverable was the algorithmic identification of the shortest synthesis pathway. This was implemented using the `reactionGraph` class (`reactionGraph.py`), which constructs a directed graph where nodes are molecules and edges represent reagent-driven transformations.

The graph search implementation had to align with the chemical strategy of retrosynthesis, the `find_path` method utilized a Breadth-First Search (BFS) algorithm on the reverse graph (where edges point (Products $\rightarrow$ Reactants)). This approach implicitly models the problem as finding the minimum number of steps required to reach any available starting material from the target molecule, effectively providing the shortest synthesis route in terms of step count. The BFS method guaranteed that the first path found was optimal for a unit-weighted graph, successfully addressing the need to return a concise and chemically reasonable route to the user.

The system successfully processed the concatenated SMIRKS input (`CCO > H2SO4 > C=C; C=C > mCPBA > OCC; OCC > HBr > BrCC; BrCC > AgOAc > OCC`), generating a complete five-step synthetic pathway:

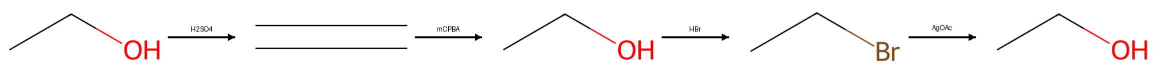

Figure 2: Visualization of a multi-step synthesis pathway. The system successfully rendered a 5-step reaction sequence, demonstrating its capability to handle complex synthetic routes and provide clear visual logic for each transformation.

This visualization, generated by the combined pipeline, verified the system's capacity to:

1. Parse and link sequential reactions defined by SMIRKS notation.
2. Track intermediate compounds (nodes) across steps.
3. Display the complete pathway as a single, interpretable image, thereby transforming the abstract algorithmic search result into a concrete pedagogical tool.

The use of an adjacency list structure (`defaultdict(list)`) in the `reactionGraph` further ensured that the graph building process from either the large dataset or curated textbook reactions was efficient and scalable, laying the groundwork for future expansions into larger reaction networks and more complex synthesis planning scenarios.

6 Conclusion

This project successfully developed and implemented an interactive visualization pipeline designed to demystify the algorithmic complexity of multistep organic synthesis for students and researchers. By bridging the gap between chemical transformations and computational logic, the tool positioned synthesis not merely as a memorization task, but as a tractable, rule-driven problem.

The foundation of the system relied on encoding chemical reaction rules using the SMIRKS formalism and leveraging the RDKit cheminformatics library for reliable structure rendering. This choice enabled the translation of vast datasets, including over a million reaction SMILES entries sourced from USPTO patent literature, alongside a curated collection of textbook reactions, into machine-interpretable data structures.

The core achievement was the establishment of a graph-based synthesis planning framework. By modeling molecules as nodes and chemical transformations (reagents/conditions) as weighted edges, the system effectively captured the decision-tree structure inherent in synthesis planning. The edge weights, defined by the number of synthetic steps, allowed the system to perform efficient path-finding, demonstrating the potential for O(1) lookup time for short, efficient routes. This framing, which treats functional group compatibility as a form of constraint satisfaction and retrosynthesis as recursive backward reasoning, successfully converted the chemical challenge into a solvable algorithmic problem.

The developed Python pipeline, utilizing RDKit and PIL, successfully parsed SMIRKS notation and automatically rendered high-quality, interpretable 2D images of both individual reaction steps and full multistep pathways. This automated visualization capability provided a critical instructional aid, moving beyond static textbook depictions to offer a dynamic, traceable record of the synthetic logic.

In conclusion, this work demonstrated the feasibility and power of applying computational methodologies to organic chemistry education. By providing an explicit visual and algorithmic structure to multistep synthesis, the project laid the groundwork for future tools that could incorporate decision-tree logic and advanced AI models for retrosynthesis prediction, ultimately enhancing the teaching and practice of chemical synthesis.

References

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.

[2] Dai, H., Li, C., Coley, C. W., Dai, B., and Song, L. *Retrosynthesis prediction with conditional graph logic network*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[3] Gale, E., Lobski, L., and Zanasi, F. A categorical model for organic chemistry. *Theor. Comput. Sci. 1032*, C (Apr. 2025).

[4] Hormazabal, R., Park, C., Lee, S., Han, S., Jo, Y., Lee, J., Jo, A., Kim, S., Choo, J., Lee, M., and Lee, H. Cede: a collection of expert-curated datasets with atom-level entity annotations for optical chemical structure recognition. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2022), NIPS '22, Curran Associates Inc.

[5] Mooring, S. R., Mitchell, C. E., and Burrows, N. L. Evaluation of a flipped, large-enrollment organic chemistry course on student attitude and achievement. *Journal of Chemical Education 93*, 12 (2016), 1972–1983. Published: December 13, 2016.

[6] O'Sullivan, B., Ferguson, A., and Freuder, E. C. Boosting constraint satisfaction using decision trees. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence* (USA, 2004), ICTAI '04, IEEE Computer Society, p. 646–651.

[7] RDKit Documentation Team. Rdkit overview, 2025. Accessed: October 22, 2025.

[8] Schilter, O. T., Laino, T., and Schwaller, P. Cmd+v for chemistry: Image to chemical structure conversion directly done in the clipboard. *Applied AI Letters 5*, 1 (Jan. 2024).

[9] Shati, P., Cohen, E., and McIlraith, S. Optimal decision trees for interpretable clustering with constraints. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence* (2023), IJCAI '23.

[10] Souza, V. F., Cicalese, F., Laber, E. S., and Molinaro, M. Decision trees with short explainable rules. *Theor. Comput. Sci. 1047*, C (Aug. 2025).

[11] van der Lingen, R. Reaction smiles crd 1.37m dataset, Jan. 2025. Dataset.