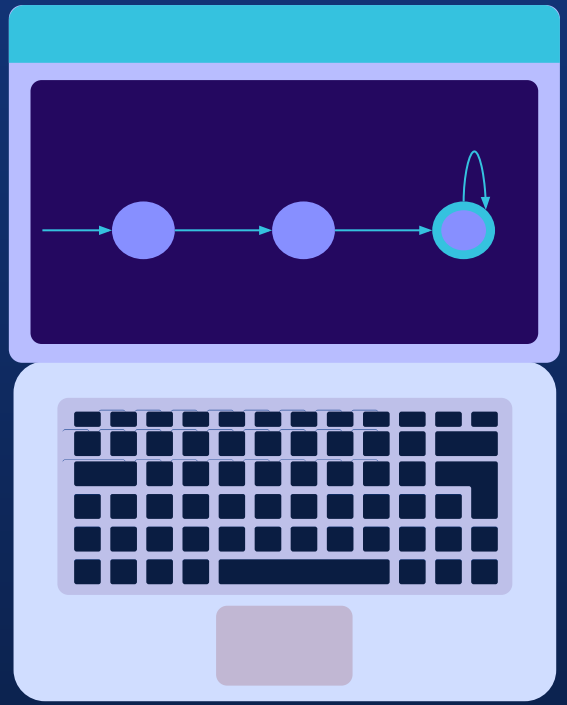
Improved Visualization for Formal Language: **Milestone 4**



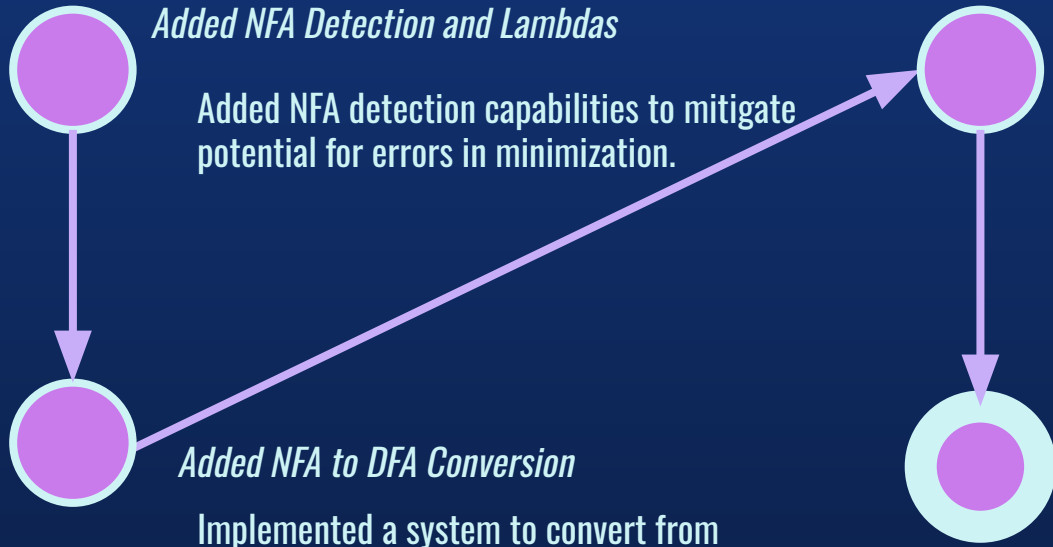
https://kmcnear2022.github.io/

Group Members: Chris Pinto-Font, Vincent Borrelli, Andrew Bastien, Keegan McNear

Milestone 4 Goals

We set out improve our visual graphing environment by adding new requested features and improving visual cohesion and design.

Milestone Four Deliverables

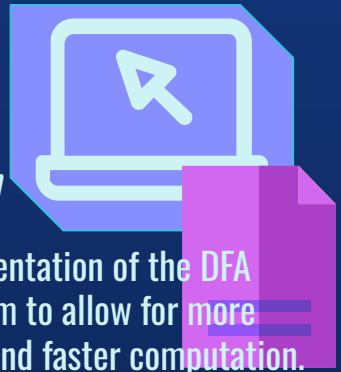


Added NFA Detection and Lambdas

Added NFA detection capabilities to mitigate potential for errors in minimization.

Refined DFA Minimization

Improved our implementation of the DFA minimization algorithm to allow for more detailed explanation and faster computation.



Added NFA to DFA Conversion

Implemented a system to convert from non-minimizable NFA's to DFA's.

Improved Animation

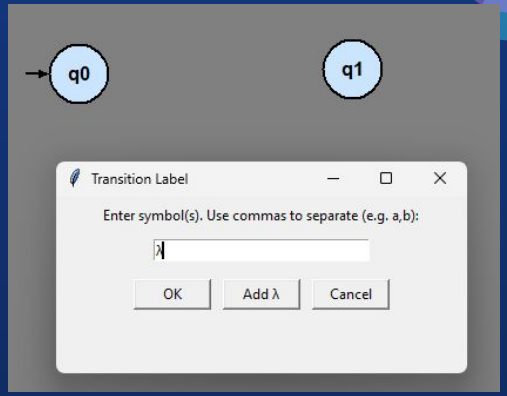
Refined and improved the animation to allow for better clarity and improved flow.

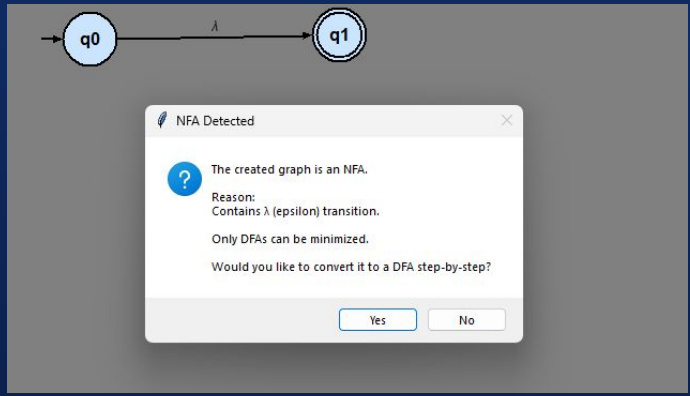
Advisor Involvement

One of the most crucial aspects of this milestone was touching base again with Dr L and redefining what the project needs to focus on. Part and parcel of that was making the decision on whether or not to accommodate NFA graphs, and if so make changes to the program in that regard

NFA inclusion and conversion.

Based upon advisor input we opted to include NFAs in an official capacity. One hurdle of this is that NFAs ***cannot*** be minimized like a DFA can. To accommodate NFA features in our program we added the capability to have lambda transitions (via button within the pop-up) and a checking system where -n which the program checks if the current graph is a DFA or an NFA, then prompts the user to convert it prior to minimization.





Code Excerpts: Refactored and Improved Animation

```
def _highlight_state(self, state: State, active: bool=False, accept: bool=False, reject:
bool=False):
        circle = self.app.state_mapping[state]
        fill = self._state_default_fill
        if reject:
            fill=self._state_reject_fill
        elif accept:
            fill=self._state_accept_fill
        elif active:
            fill=self._state_active_fill
        self.canvas.itemconfig(circle, fill=fill)
```

Highlights states in the traversal and minimization process to display traversal and changes.

Code Excerpts: Added Epsilon Closure

```
def epsilon_closure(self, state: State | UUID) -> set[State]:
        state = self._ensure_state(state)
        closure = {state}
        stack = [state]
        while stack:
            current = stack.pop()
            for t in self.transitions_from(current):
                if t.is_lambda() and t.target not in closure:
                    closure.add(t.target)
                    stack.append(t.target)
        return closure
```

Added Epsilon Closure to allow for conversion of epsilon NFA’s to DFA’s

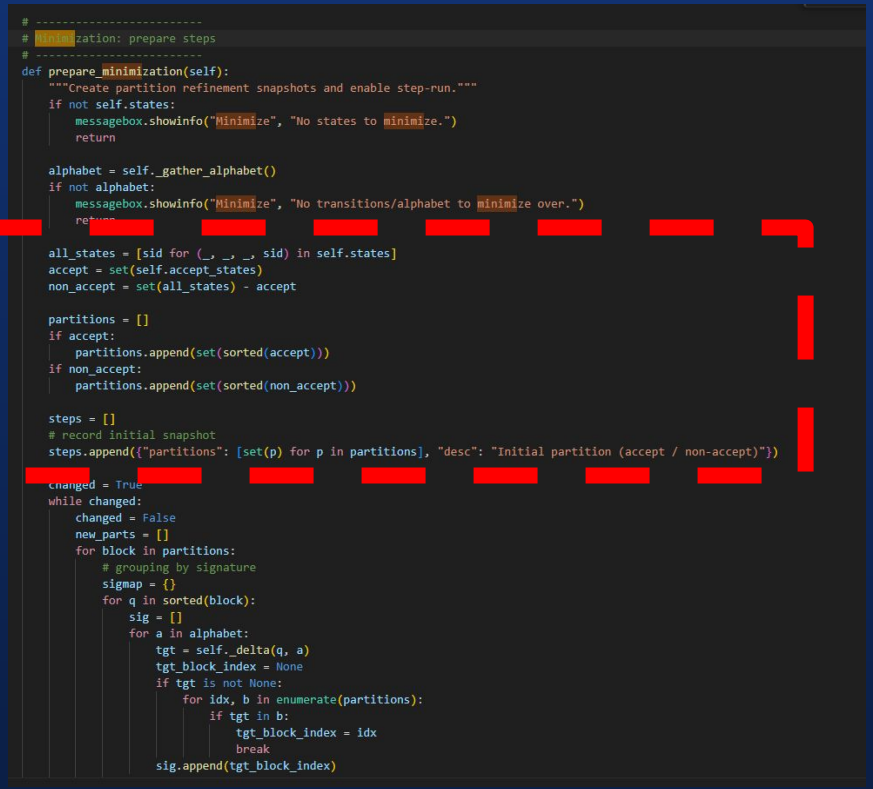
Code Excerpts: Refactored Action System

```
def perform_actions(self, actions: list[tuple[str, ...]]):
        for action in actions:
            self.perform_action(action)

    def perform_action(self, action: tuple[str, ...]):
        match action:
            case ("add_state", (id, (x, y), label)):
                pos = Positioning(x, y)
                self.add_state(id, pos, label, undoable=False)
            case ("add_symbols", (source_id, target_id, symbols)):
                state = self.machine.states[source_id]
                target = self.machine.states[target_id]
                self.add_transition(state, target, symbols, undoable=False)
            case ("delete_state", (id, (x, y), label)):
                state = self.machine.states[id]
                self.delete_state(id, undoable=False)
            case ("remove_symbols", (source_id, target_id, symbols)):
                state = self.machine.states[source_id]
                target = self.machine.states[target_id]
                self.delete_transition(state, target, symbols, undoable=False)
            case ("move_state", (id, (x, y), label)):
                state = self.machine.states[id]
                pos = Positioning(x, y)
                self.move_state(state, pos, undoable=False)
            case ("rename_state", (id, old, new_label)):
                state = self.machine.states[id]
                self.rename_state(state, new_label, undoable=False)
```

Refactored action system allows for more efficient processing for lowered response time and lighter load.

Code Excerpts: Refactored and Improved Animation



```
# ------------------------
# Minimization: prepare steps
# ------------------------
def prepare_minimization(self):
    """Create partition refinement snapshots and enable step-run."""
    if not self.states:
        messagebox.showinfo("Minimize", "No states to minimize.")
        return

    alphabet = self._gather_alphabet()
    if not alphabet:
        messagebox.showinfo("Minimize", "No transitions/alphabet to minimize over.")
        return

    all_states = [sid for (_, _, _, sid) in self.states]
    accept = set(self.accept_states)
    non_accept = set(all_states) - accept

    partitions = []
    if accept:
        partitions.append(set(sorted(accept)))
    if non_accept:
        partitions.append(set(sorted(non_accept)))

    steps = []
    # record initial snapshot
    steps.append({"partitions": [set(p) for p in partitions], "desc": "Initial partition (accept / non-accept)"})

    changed = True
    while changed:
        changed = False
        new_parts = []
        for block in partitions:
            # grouping by signature
            sigmap = {}
            for q in sorted(block):
                sig = []
                for a in alphabet:
                    tgt = self._delta(q, a)
                    tgt_block_index = None
                    if tgt is not None:
                        for idx, b in enumerate(partitions):
                            if tgt in b:
                                tgt_block_index = idx
                                break
                    sig.append(tgt_block_index)
```

Prepares for Minimization Process.

Encompasses the very first partition used in the minimization algorithm, dividing states into accepting states and non-accepting states.

Milestone 4 Progress

Task	Completion %	Chris	Vincent	Andrew	Keegan	To Do
NFA detection + Lambda	100%	50%	50%	0%	0%	Nothing
NFA to DFA conversion	90%	90%	0%	0%	0%	Must refine the logic to improve accuracy and add a separate conversion from minimization
DFA Minimization	90%	45%	45%	0%	0%	Refine to make sure 100% accurate
Improved animation	80%	0%	0%	0%	80%	Animation needs a “wow” factor
Refactorization	100%	0%	0%	100%	0%	Nothing

Milestone 5 Plan

Task	Chris	Vincent	Andrew	Keegan
NFA dead state and bug fixing	Logic implementation and researcher	Bug Fixer/Code Contributor and designer	Co-Lead coder and development head	Co-Lead coder and development head
Canvas GUI upgrade with quality of life features	Bug Fixer/Code Contributor and researcher	Bug Fixer/Code Contributor and researcher	Co-Lead coder and development head	Co-Lead coder and development head
Tutorial mode	Bug Fixer/Code Contributor and researcher	Bug Fixer/Code Contributor and researcher	Co-Lead coder and development head	Co-Lead coder and development head
String input converted into a DFA	Logic writer, DFA logic consultant, and bug tester	Logic writer, DFA logic consultant, and bug tester	Co-Lead code side implementor	Co-Lead code side implementor
User feedback	Bug Fixer/Code Contributor and designer	Bug Fixer/Code Contributor and designer	Co-Lead code side implementor	Co-Lead code side implementor

Questions?

Visit Our Site

https://kmcnear2022.github.io/