

# EECS4302 Compilers and Interpreters

## Fall 2026

### Digest of Lecture 7, October 1

#### NFA Processing, $\varepsilon$ -Closure, and RE-to- $\varepsilon$ -NFA Conversion

Jackie Wang

##### HOW TO USE THIS DIGEST

Use this digest as a **roadmap and checklist**. It highlights the main ideas, important examples, common pitfalls, practice targets, and where to return in the course materials. It does **not** replace attending or reviewing the session, using the recording when needed, studying Jackie's annotated iPad notes and other official course materials, completing assigned work, or practising independently. Use it to identify what you understand, what you need to revisit, and what you should be able to do on your own.

If time is limited, start with the sections most relevant to your current review and return to the rest later; selective use is better than skipping the digest because it looks long.

This is an AI-assisted study roadmap reviewed by Jackie and grounded in the session transcript and Jackie's annotated iPad notes. Like any course document, it may still contain an error or omission. If something seems technically doubtful or inconsistent with the original course materials, return to those materials and ask Jackie for clarification rather than treating the digest as final authority.

##### TABLE OF CONTENTS

|                                     |   |
|-------------------------------------|---|
| Main Ideas                          | 2 |
| Roadmap                             | 2 |
| Key Terms, Notation, and Structures | 5 |
| Key Examples and Demonstrations     | 6 |
| Common Pitfalls and Misconceptions  | 6 |
| What You Should Be Able to Do       | 7 |
| Suggested Practice                  | 8 |
| Where to Review                     | 8 |
| Closing Reminder                    | 9 |

## Main Ideas

- An NFA does not follow a single current state. After each input symbol, it may be in a **set of possible states**; the input is accepted if the final reachable set contains at least one accepting state.
- An  $\varepsilon$ -NFA extends the same nondeterministic model with  $\varepsilon$ -transitions, which consume no input. These transitions make it easier to connect smaller automata when modeling concatenation and union.
- $\varepsilon$ -closure is the key operation for reasoning about an  $\varepsilon$ -NFA. It always contains the starting state itself and then recursively includes every state reachable using only  $\varepsilon$ -transitions.
- Processing a character in an  $\varepsilon$ -NFA has two stages: first follow transitions on the input symbol from all currently possible states; then take the  $\varepsilon$ -closure of the resulting states.
- The conversion from a regular expression to an equivalent  $\varepsilon$ -NFA is a systematic recursive construction. Jackie emphasized following the stated mechanism faithfully, including the explicit  $\varepsilon$ -transitions in the union, concatenation, and Kleene-star cases.
- Assignment 1 is due October 7, and the regular-expression-to-  $\varepsilon$ -NFA conversion is one of the transformations students must implement. Jackie also emphasized practising the construction and following the stated mechanism faithfully.

## Roadmap

### 1. Logistics, Assignment 1, and the next milestones

Jackie opened with a reminder that Assignment 1 is due October 7 and urged anyone who had not started to begin soon. He also said that the project and Assignment 2 would be released before Thursday's class the following week, with a short briefing at the beginning of that class. The intention is to give students useful time over reading week. Jackie described the Programming Test as being around November 5 and said that studying the ANTLR4 tutorial and working through Assignment 2 will be part of preparing for it.

### 2. NFA processing means tracking a set of possible states

The session first returned to the NFA for binary strings that end in 01. Jackie traced the input

00101

from the start state  $q_0$ . After an input symbol, nondeterminism can create more than one possible current state. A branch that has no valid transition simply gets stuck; the other branches continue.

At the end of the trace, the possible states are  $\{q_0, q_2\}$ , while the set of accepting states is  $F = \{q_2\}$ . Therefore

$$F \cap \{q_0, q_2\} \neq \emptyset,$$

so the string is accepted. Jackie's central point was the processing model: after *every* input character, maintain the complete set of possible states rather than trying to choose a single path.

Jackie used the “alternate universe” idea from *Back to the Future* as an intuition for this branching behavior. The analogy is only a memory aid; the actual acceptance rule is the set-of-states calculation above.

### 3. Why $\varepsilon$ -transitions are useful

An  $\varepsilon$ -NFA allows a transition without consuming another symbol from the input. Jackie used two compositional examples in Jackie's annotated iPad notes:

- **Concatenation:** build one automaton for a first pattern  $x$  and another for a second pattern  $y$ , turn the first component's accepting states into ordinary states in the combined machine, and connect them to the second component's start state using  $\varepsilon$ -transitions.
- **Union:** keep the two component machines independent and add a new start state with  $\varepsilon$ -transitions to both component start states.

These constructions make the combined structure easier to design without giving  $\varepsilon$ -NFAs more expressive power than ordinary NFAs or DFAs.

### 4. Tracing a floating-point $\varepsilon$ -NFA

Jackie's annotated iPad notes then use an  $\varepsilon$ -NFA for a simplified floating-point-number language. A string has an optional sign, decimal digits before the dot, and decimal digits after the dot, with the important constraint that the digit strings on the two sides of the dot cannot both be empty.

The class traced examples including  $.23$  and  $+46.$ . The point was to read the formal language constraint together with the diagram: the automaton must reject a bare dot and a signed bare dot, while still allowing one side of the dot to have no digits. A student also asked whether every  $\varepsilon$ -transition shown was strictly necessary. Jackie clarified that the illustrated machine is correct but need not be minimal; minimization is a separate concern.

### 5. $\varepsilon$ -closure: definition and recursive computation

For a state  $q$ ,  $\text{ECLOSE}(q)$  is the set of states reachable from  $q$  using only  $\varepsilon$ -transitions, including  $q$  itself. Jackie emphasized the base case:

$$q \in \text{ECLOSE}(q).$$

Then, for each state reached by one  $\varepsilon$ -transition, continue computing its  $\varepsilon$ -closure recursively until no further  $\varepsilon$ -transition can be followed.

In the worked diagram on page 6 of Jackie's annotated iPad notes,

$$\text{ECLOSE}(q_0) = \{q_0, q_1, q_2, q_3, q_5\}.$$

A state with no outgoing  $\varepsilon$ -transition still has a non-empty closure: its closure contains at least the state itself.

## 6. Processing a character in an $\varepsilon$ -NFA

The next example began tracing the input 5.6. Before reading the first character, start from the  $\varepsilon$ -closure of the start state rather than from the start state alone. In the diagram on page 7 of Jackie's annotated iPad notes,

$$\text{ECLOSE}(q_0) = \{q_0, q_1\}.$$

To process the first character 5, apply the ordinary transition function from both possible current states and take the union of the results. The lecture obtains

$$\delta(q_0, 5) \cup \delta(q_1, 5) = \emptyset \cup \{q_1, q_4\} = \{q_1, q_4\}.$$

That is only the first phase. The second phase is to compute

$$\text{ECLOSE}(q_1) \cup \text{ECLOSE}(q_4),$$

which gives the new current-state set  $\{q_1, q_4\}$  in this example. Jackie left the rest of the trace for .6 as an exercise.

## 7. The regular-expression-to- $\varepsilon$ -NFA construction

The central construction of the second half of the session was a recursive method for turning a regular expression into an equivalent  $\varepsilon$ -NFA. Jackie highlighted structural constraints for the construction: the resulting machine has one start state and exactly one accepting state, with no transition entering the start state and no transition leaving the accepting state.

The three base cases are shown on page 8 of Jackie's annotated iPad notes:

- $\varepsilon$ : start  $\xrightarrow{\varepsilon}$  accept;
- $\emptyset$ : a start state and an accept state with no connecting transition, so no string is accepted;
- $a \in \Sigma$ : start  $\xrightarrow{a}$  accept.

Given already-constructed machines for regular expressions  $R$  and  $S$ , the recursive cases are:

- **Union**  $R + S$ : introduce a new start state that branches by  $\varepsilon$  to the two component starts, and a new accept state reached by  $\varepsilon$  from the two component accepts.
- **Concatenation**  $RS$ : connect the accept state of the  $R$  machine to the start state of the  $S$  machine with an explicit  $\varepsilon$ -transition. The overall start comes from the leftmost component, and the overall accept comes from the rightmost component.
- **Kleene star**  $R^*$ : introduce new start and accept states, allow an  $\varepsilon$  path for zero repetitions, enter the  $R$  machine by  $\varepsilon$ , leave it by  $\varepsilon$ , and add the  $\varepsilon$  loop-back needed for repeated executions of  $R$ .

Jackie stressed that the assessment version may explicitly ask students to follow this mechanism. An alternative automaton can recognize the same language and still be mathematically correct, but it may not be faithful to the required construction.

## 8. Applying the construction to concrete regular expressions

The first worked construction builds the machine for

$$(0 + 1)^*.$$

The decomposition follows the syntax: build the base machines for 0 and 1, combine them using the union construction for  $0 + 1$ , then wrap the result using the Kleene-star construction. Page 9 of Jackie's annotated iPad notes shows the nested structure visually.

The lecture then extended the same mechanism to

$$(0 + 1)^*1(0 + 1).$$

The three component machines are connected by the concatenation rule, including explicit  $\varepsilon$ -transitions between components. Page 10 of Jackie's annotated iPad notes also marks the overall start as the start state of the leftmost component and the overall accept as the accept state of the rightmost component.

## 9. Look ahead

Jackie said that the next Tuesday session would spend roughly the first 10–15 minutes on the critical steps for converting an  $\varepsilon$ -NFA to a DFA. The course would then make a short digression into object-oriented design patterns, including polymorphism, dynamic binding, Composite, and Visitor, to prepare for the ANTLR4 tutorial work before reading week.

## Key Terms, Notation, and Structures

|                                            |                                                                                                                                                |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NFA current-state set</b>               | The set of every state the NFA could occupy after the input read so far. A single input symbol can produce multiple possible resulting states. |
| $F$                                        | The set of accepting states. An NFA accepts when its final reachable state set has non-empty intersection with $F$ .                           |
| <b><math>\varepsilon</math>-transition</b> | A transition that changes state without consuming an input symbol.                                                                             |
| $\text{ECLOSE}(q)$                         | The set containing $q$ and every state reachable from $q$ using only $\varepsilon$ -transitions.                                               |
| $\delta(q, a)$                             | The transition result from state $q$ on input symbol $a$ . For an NFA or $\varepsilon$ -NFA, the result may be a set of states.                |
| <b>Base cases for RE conversion</b>        | Separate constructions for $\varepsilon$ , $\emptyset$ , and a single alphabet symbol $a \in \Sigma$ .                                         |

### Recursive cases for RE conversion

Structural constructions for union  $R + S$ , concatenation  $RS$ , and Kleene star  $R^*$  using already-constructed machines for  $R$  and  $S$ .

## Key Examples and Demonstrations

### NFA trace for 00101

**Setup:** Use the three-state NFA that recognizes binary strings ending in 01.

**Main point:** Maintain a set of possible states after each character. Some branches can get stuck while other branches continue. The final reachable set is  $\{q_0, q_2\}$ , which intersects  $F = \{q_2\}$ , so the string is accepted.

**What to practise:** Reproduce the trace from page 3 of Jackie's annotated iPad notes without choosing a single "best" branch.

### Floating-point $\varepsilon$ -NFA

**Setup:** Use the diagram on page 5 of Jackie's annotated iPad notes and the constraint that the digit strings before and after the dot are not both empty.

**Main point:** .23 and +46. are accepted. The example also distinguishes correctness from minimization: a valid  $\varepsilon$ -NFA can contain transitions that a later optimization could remove.

**What to practise:** Trace a candidate string against both the formal constraint and the diagram, rather than checking the picture alone.

### $\varepsilon$ -closure derivation

**Setup:** Starting at  $q_0$  in the page-6 diagram, recursively follow only  $\varepsilon$ -transitions.

**Main point:**

$$\text{ECLOSE}(q_0) = \{q_0, q_1, q_2, q_3, q_5\}.$$

The base state is always included, even when there is no outgoing  $\varepsilon$ -transition.

**What to practise:** Expand one branch at a time until every branch reaches a state with no further  $\varepsilon$ -transition.

### Regular expression to $\varepsilon$ -NFA

**Setup:** Follow the page-8 construction rules, then revisit the diagrams for  $(0 + 1)^*$  and  $(0 + 1)^*1(0 + 1)$  on pages 9–10.

**Main point:** The construction is compositional. Build the smaller machines first, then combine them according to the operator at the next level of the regular expression.

**What to practise:** Identify the decomposition of the regular expression before drawing any transitions, and preserve the explicit  $\varepsilon$ -connections required by the mechanism.

## Common Pitfalls and Misconceptions

- **Treating an NFA like a DFA.** Do not keep only one current state. After each symbol, compute the complete set of possible resulting states.

- **Thinking a stuck branch means immediate rejection.** A stuck NFA branch is discarded, but other branches may still lead to acceptance. Rejection occurs only when no accepting possibility remains after the full input.
- **Returning the empty set for an  $\varepsilon$ -closure with no outgoing  $\varepsilon$  edge.** The state itself is always in its own closure.
- **Skipping the second phase of  $\varepsilon$ -NFA processing.** After following transitions on an input symbol, compute the  $\varepsilon$ -closure of every resulting state before reading the next symbol.
- **Ignoring a constraint from the language definition.** In the floating-point example, the digit strings before and after the dot cannot both be empty; checking only a visible path without the full specification can admit invalid strings.
- **Using a regular expression such as  $R$  as an automaton transition label.** Jackie explicitly rejected this shortcut: a transition label must be an alphabet symbol or  $\varepsilon$ , not an entire pattern.
- **Replacing the taught concatenation construction with an intuitive shortcut.** When asked to follow the lecture/slide mechanism, use the explicit  $\varepsilon$ -transition between the component machines. A different equivalent automaton may be correct as a language recognizer but not faithful to the required construction.

## What You Should Be Able to Do

After this session, you should be able to:

- trace an NFA on a concrete input by maintaining the complete set of possible states after each symbol;
- decide NFA acceptance using the intersection of the final reachable state set with the accepting set  $F$ ;
- explain how  $\varepsilon$ -transitions support nondeterministic modeling without consuming input;
- compute  $\text{ECLOSE}(q)$  from a diagram and explain why the starting state itself is always included;
- process one or more input characters in an  $\varepsilon$ -NFA using the transition-then- $\varepsilon$ -closure procedure;
- distinguish a correct automaton from a minimal automaton;
- construct the base-case  $\varepsilon$ -NFAs for  $\varepsilon$ ,  $\emptyset$ , and  $a \in \Sigma$ ;
- combine component machines correctly for  $R + S$ ,  $RS$ , and  $R^*$ ;
- reproduce the construction strategy for  $(0+1)^*$  and explain how it extends to  $(0+1)^*1(0+1)$ ; and

- explain why following the specified RE-to- $\varepsilon$ -NFA mechanism matters when an assessment explicitly asks for that mechanism.

## Suggested Practice

1. Reproduce the page-3 trace of 00101 on the NFA for strings ending in 01. At each character, write the entire resulting state set before moving on.
2. Complete the page-7 processing exercise for the remaining characters .6 in the input 5.6. For each character, separate the ordinary transition step from the subsequent  $\varepsilon$ -closure step.
3. Re-derive  $\text{ECLOSE}(q_0)$  on page 6 without looking at the final answer, and check that you stop only when every  $\varepsilon$  branch has been exhausted.
4. Practise the regular-expression-to- $\varepsilon$ -NFA mechanism on a few additional regular expressions, as Jackie recommended. Start from the relevant base machines and apply only the union, concatenation, and star constructions from the session.
5. Reconstruct  $(0+1)^*$  and then  $(0+1)^*1(0+1)$  from pages 9–10 without copying the finished diagram. Check the overall start/accept states and every explicit  $\varepsilon$ -connection.
6. Begin or continue Assignment 1 now. In particular, connect the formal RE-to- $\varepsilon$ -NFA construction from this lecture to the transformation you are required to implement before the October 7 deadline.
7. If useful before the next lecture, look ahead at the course material on converting an  $\varepsilon$ -NFA to a DFA so that the critical steps Jackie highlights next Tuesday have context.

## Where to Review

- **Jackie's annotated iPad notes, page 2:** Assignment 1 due date and self-study reminder.
- **Page 3:** NFA behavior, the 00101 trace, the accepting set  $F = \{q_2\}$ , and the non-empty-intersection acceptance check.
- **Page 4:** motivation for  $\varepsilon$ -NFAs through concatenation and union of component automata.
- **Page 5:** simplified floating-point language and the worked  $\varepsilon$ -NFA used for string traces.
- **Page 6:**  $\varepsilon$ -closure definition and the derivation of  $\text{ECLOSE}(q_0) = \{q_0, q_1, q_2, q_3, q_5\}$ .
- **Page 7:** processing input 5.6 and the transition-then-closure workflow; the remaining characters are left as practice.
- **Page 8:** the base and recursive cases for converting a regular expression to an  $\varepsilon$ -NFA.

- **Pages 9–10:** the worked constructions for  $(0 + 1)^*$  and  $(0 + 1)^*1(0 + 1)$ , including the overall start/accept-state annotations.
- **Session recording:** return to the spoken explanation if you need the reasoning behind the NFA “alternate universe” analogy, the assessment warning about following the stated construction mechanism, or the discussion of upcoming Assignment 2/project and ANTLR4 preparation.

## Closing Reminder

Use this digest as a roadmap for active review, not as a substitute for the learning process itself. Attend Jackie’s in-person sessions and pay attention to how he develops ideas and works through the important reasoning step by step. If a transition, explanation, example, or reasoning step remains unclear afterwards, return to the recording to fill that gap.

For structured review, start with Jackie’s annotated iPad notes because they most closely reflect the actual flow and development of the class. Work through them in manageable logical units. After each logical unit, study the corresponding part of the presentation slides alongside it. Jackie does not normally teach by reading the slides line by line, so the slides may contain relevant supporting material that was not explicitly read aloud.

Finally, choose a suitable selection of the important reasoning from this session and work through it again on your own: an NFA state-set trace, an  $\varepsilon$ -closure derivation, one or more steps of  $\varepsilon$ -NFA string processing, and at least one regular-expression-to- $\varepsilon$ -NFA construction. The goal is not to memorize Jackie’s recorded steps, but to determine whether you can reproduce the reasoning from your own understanding and explain why the important steps work.