



Relating REs, NFAs, DFAs, CFGs

CSE 311: Foundations of
Computing I
Lecture 23

Announcements

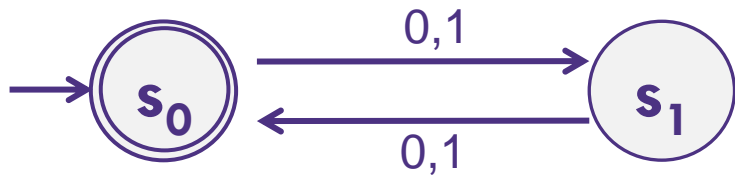
- HW6 solutions are at the front, grades to be published soon
- HW7 due on Wednesday at 11:59 pm on Grin
 - No Late Days permitted
- Final Exam is this Thursday and Friday
 - Information posted on Exams page
 - Optional review session tomorrow from 3:00 – 4:20pm in DEM 104. Will be recorded on Panopto.

Recall: REs & CFGs

- Regular Languages are languages that can be matched by a Regular Expression.
For Example, $(0 \cup 1)^*11(0 \cup 1)^*$ matches all binary strings containing 11.
- Context-Free Languages are languages that can be generated by a Context-Free Grammar.
For Example, $S \rightarrow 0S \mid S1 \mid \varepsilon$ generates all binary strings of the form $\{0^n1^n : n \geq 0\}$

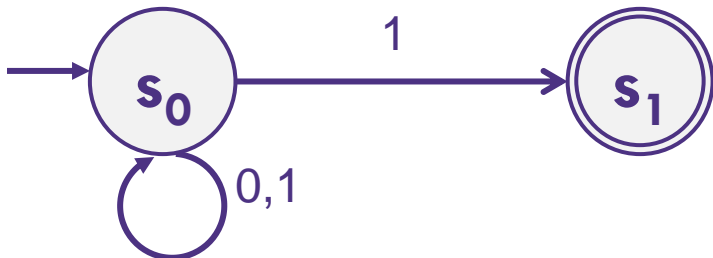
Recall: DFAs and NFAs

- Deterministic Finite Automata are one model of computation. DFAs require exactly one outgoing edge per state per character in Σ . Accepts a string iff there is a path to an accept state. For example:



Recognizes binary strings of even length

- Nondeterministic Finite Automata are another model of computation. NFAs allow for multiple or no edges per state per character in Σ . Accepts a string iff there is a path to an accept state. For example:



Recognizes binary strings ending in 1

The story so far...

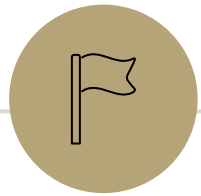
REs

CFGs

DFAs

=

NFAs

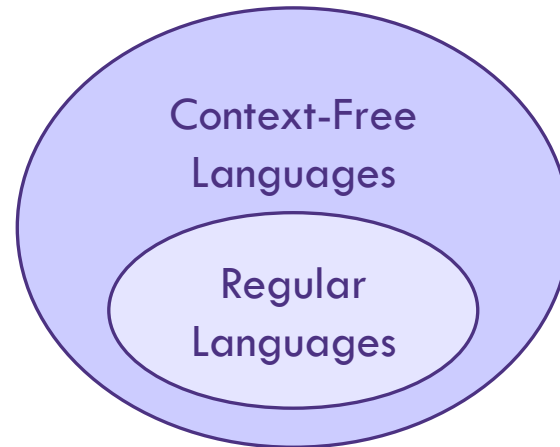


REs \subseteq CFGs

REs and CFGs

Theorem 1:

For any language A matched by a Regular Expression, there is a CFG that generates A .



Proof Idea: Structural Induction! Show that claim holds for all Basis Steps and Recursive Steps of REs.

Recall: Regular Expressions

Basis Step:

- ε is a regular expression
- a is a regular expression for any $a \in \Sigma$

Recursive Step: If A and B are regular expressions, then...

- $A \cup B$ is a regular expression
- AB is a regular expression
- A^* is a regular expression

Proof of Theorem 1

Basis Steps:

ε is a regular expression

a is a regular expression for any $a \in \Sigma$

1. Let $P(R)$ be "For a Regular Expression R , there exists a CFG that generates the same language". We prove $P(R)$ for all REs R by structural induction.

2. Base Cases:

$R = \varepsilon$: ε is generated by the CFG $S \rightarrow \varepsilon$.

$R = a$: a is generated by the CFG $S \rightarrow a$.

Proof of Theorem 1

Recursive Step: If A, B are REs...

- $A \cup B$ is a RE
- AB is a RE
- A^* is a RE

3. IH: Let A, B be arbitrary REs. Suppose there exists a CFG with start symbol S_1 that generates the language matched by A . Suppose there exists a CFG with start symbol S_2 that generates the language matched by B .
4. IS:
 $R = A \cup B$. The language of the RE $A \cup B$ is generated by the CFG $S \rightarrow S_1 \mid S_2$.
 $R = AB$. The language of the RE AB is generated by the CFG $S \rightarrow S_1 S_2$.
 $R = A^*$. The language of the RE A^* is generated by the CFG $S \rightarrow S_1 S \mid \varepsilon$.
5. Conclusion: (omitted)

Example of RE to CFG Construction

The proof of Theorem 1 gives us a process to write every RE as a CFG.
For example, write the RE $(a \cup b)^*c$ as a CFG using the proof construction.

1. To write a , we can use the CFG

$$A \rightarrow a$$

2. To write b , we can use the CFG

$$B \rightarrow b$$

3. To write c , we can use the CFG

$$C \rightarrow c$$

4. To write $a \cup b$, we can use the CFG

$$D \rightarrow A \mid B$$

$$A \rightarrow a$$

$$B \rightarrow b$$

5. To write $(a \cup b)^*$, we can use the CFG

$$E \rightarrow DE \mid \varepsilon$$

$$D \rightarrow A \mid B$$

$$A \rightarrow a$$

$$B \rightarrow b$$

6. To write $(a \cup b)^*c$, we can use the CFG

$$F \rightarrow EC$$

$$E \rightarrow DE \mid \varepsilon$$

$$D \rightarrow A \mid B$$

$$A \rightarrow a$$

$$B \rightarrow b$$

$$C \rightarrow c$$

Example of RE to CFG Construction

The proof of Theorem 1 gives us a process to write every RE as a CFG.
For example, write the RE $(a \cup b)^*c$ as a CFG using the proof construction.

$F \rightarrow EC$

$E \rightarrow DE \mid \varepsilon$

$D \rightarrow A \mid B$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow c$

Example of RE to CFG Construction

Note: This construction will not necessarily give us the simplest CFG.

For example, for $(a \cup b)^*c$, the construction gave:

$$F \rightarrow EC$$
$$E \rightarrow DE \mid \varepsilon$$
$$D \rightarrow A \mid B$$
$$A \rightarrow a$$
$$B \rightarrow b$$
$$C \rightarrow c$$

But this CFG would also work:

$$S \rightarrow AS \mid c$$
$$A \rightarrow a \mid b$$

The story so far...

REs

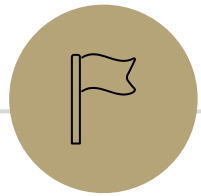
\supseteq

CFGs

DFAs

$=$

NFAs



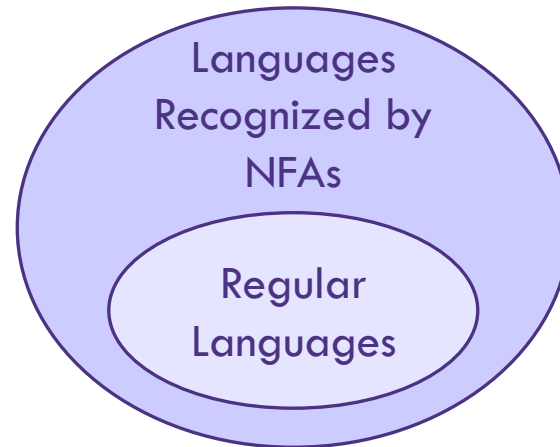
REs \subseteq NFAs



REs and NFAs

Theorem 2:

For any language A matched by a Regular Expression, there is an NFA that recognizes A .



Proof Idea: Structural Induction again! Show that claim holds for all Basis Steps and Recursive Steps of REs.

Proof Sketch of Theorem 2

Basis Steps:

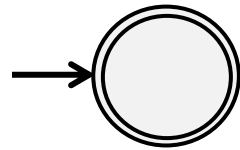
ε is a regular expression

a is a regular expression for any $a \in \Sigma$

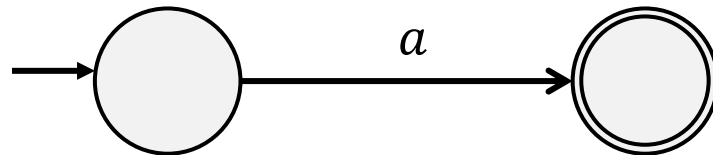
1. Let $P(R)$ be "For a Regular Expression R , there exists an NFA that recognizes the same language". We prove $P(R)$ for all REs R by structural induction.

2. Base Cases:

$R = \varepsilon$



$R = a$

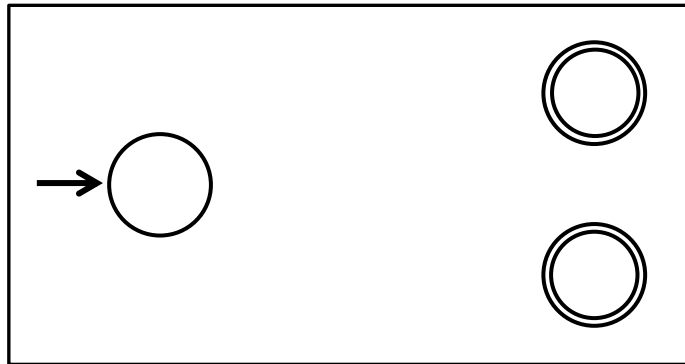


Proof Sketch of Theorem 2

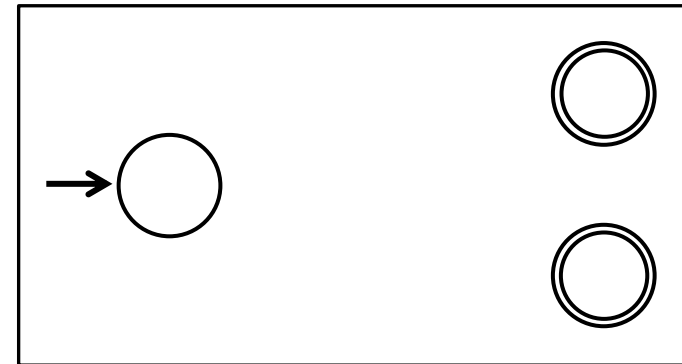
Recursive Step: If A, B are REs...

- $A \cup B$ is a RE
- AB is a RE
- A^* is a RE

3. IH: Suppose there for some arbitrary REs A, B there exists NFAs N_A and N_B that recognize the language matched by A and B respectively.



N_A



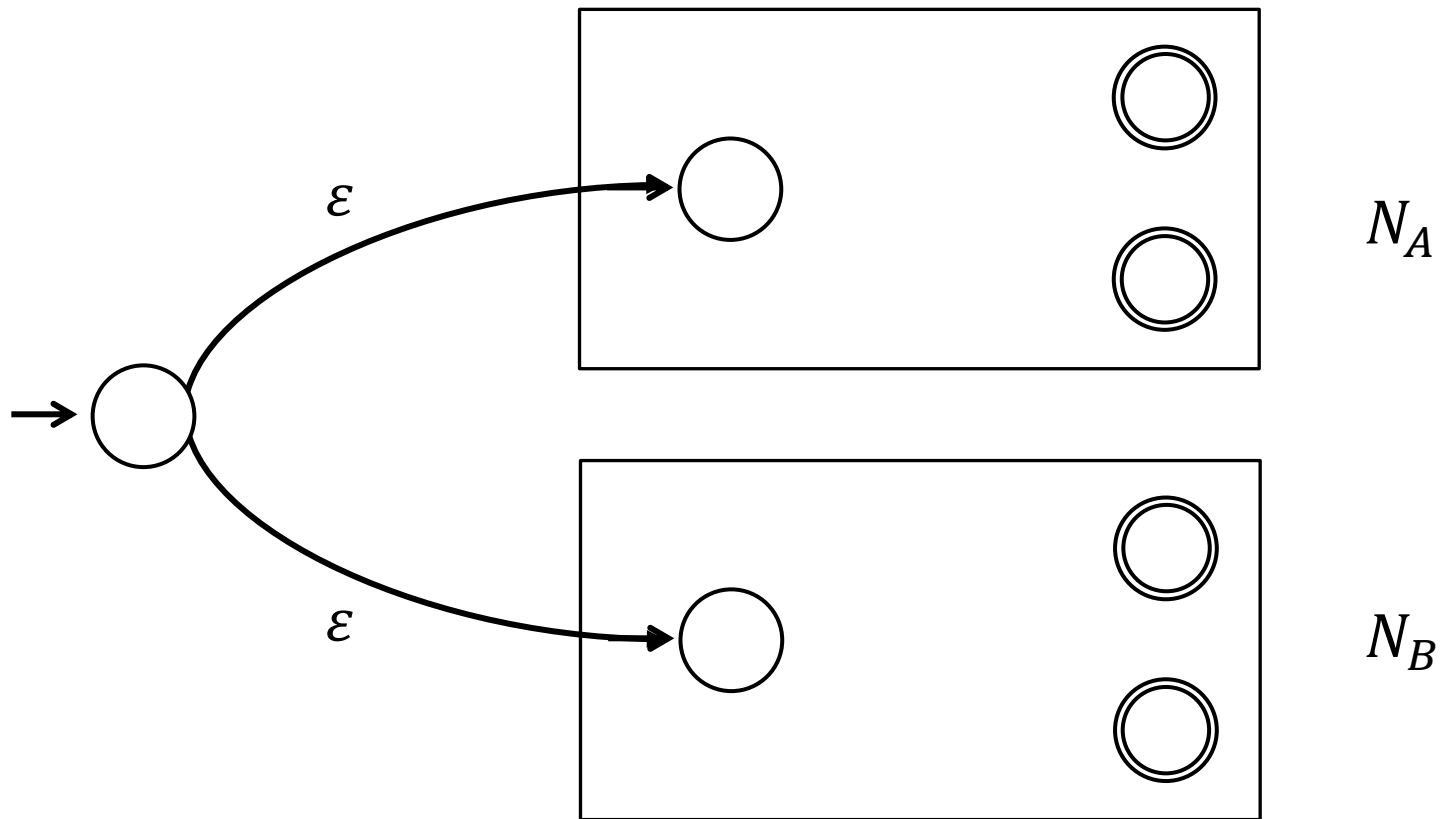
N_B

Proof Sketch of Theorem 2

Recursive Step: If A, B are REs...

- $A \cup B$ is a RE
- AB is a RE
- A^* is a RE

4. IS: The RE $A \cup B$ can be matched by the following NFA:

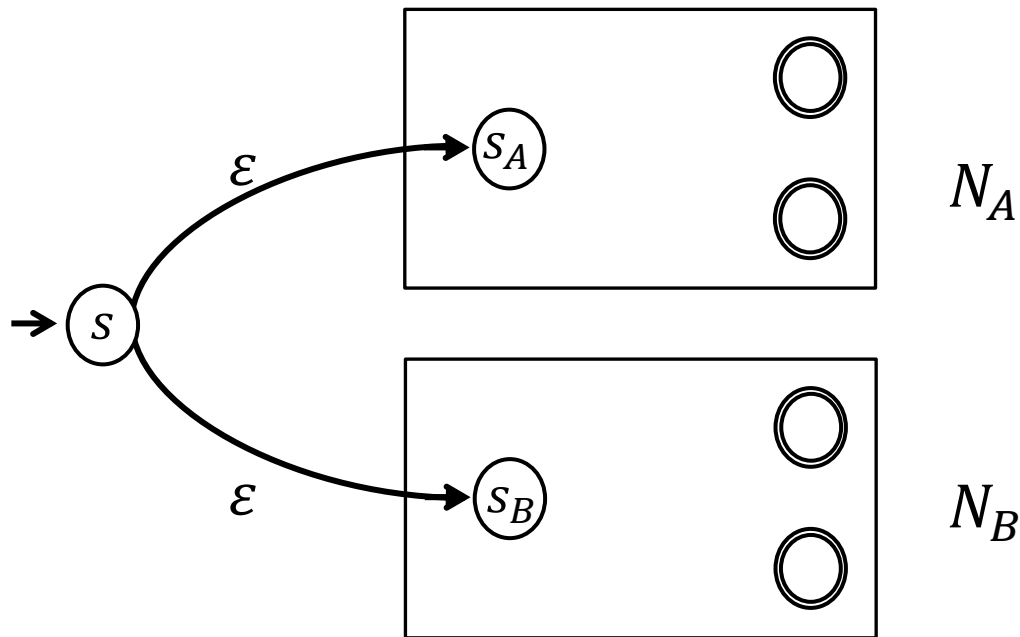


Proof Sketch of Theorem 2

Recursive Step: If A, B are REs...

- $A \cup B$ is a RE
- AB is a RE
- A^* is a RE

4. IS: The RE $A \cup B$ can be matched by the following NFA:



Any string matched by $A \cup B$ is accepted by the new NFA

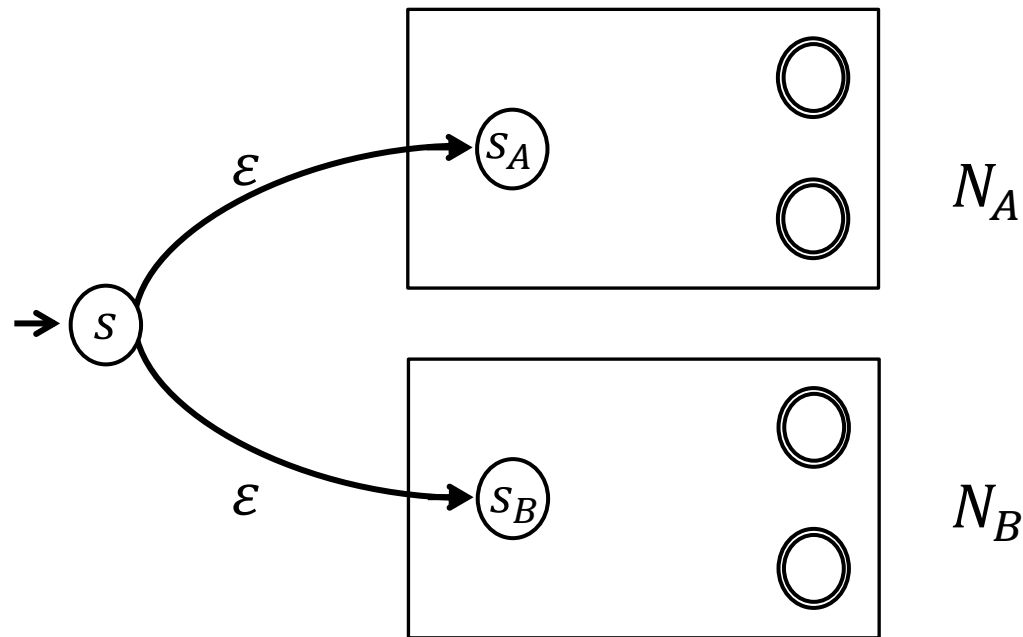
- Suppose x matches $A \cup B$
- Then x matches A or x matches B . Assume without loss of generality that x matches A .
- Then there is a path labelled x from s_A to an accept state in N_A .
- Then there is a path labelled x from s to an accept state in the new NFA.
- So x is accepted by the new NFA.

Proof Sketch of Theorem 2

Recursive Step: If A, B are REs...

- $A \cup B$ is a RE
- AB is a RE
- A^* is a RE

4. IS: The RE $A \cup B$ can be matched by the following NFA:



Any string accepted by the new NFA matches $A \cup B$.

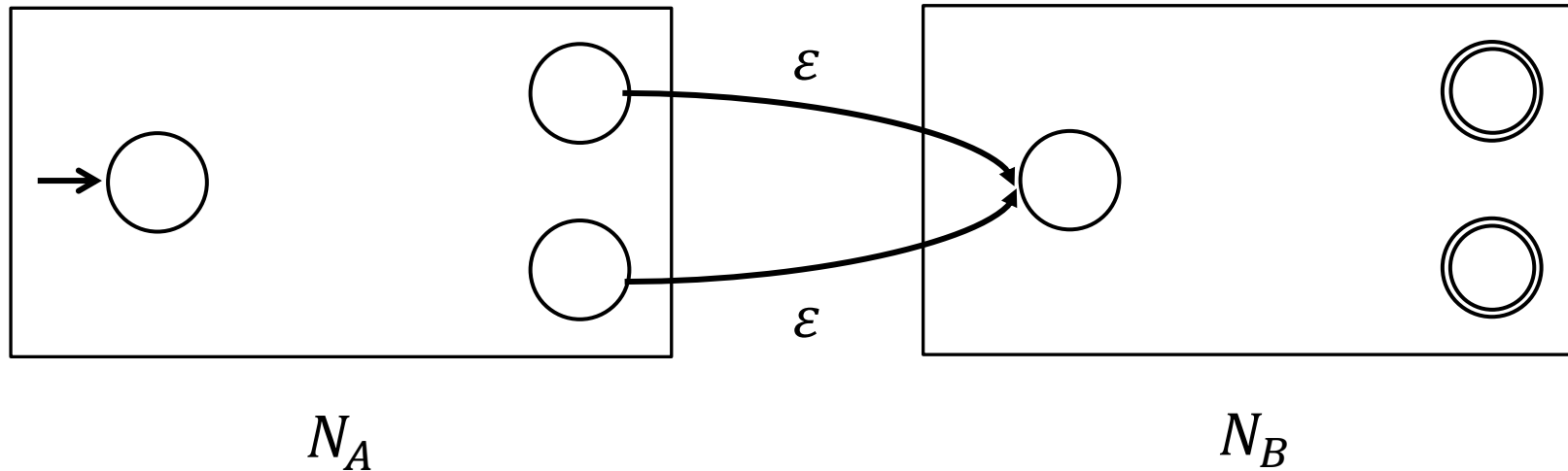
- Suppose x is accepted by the new NFA.
- Then there is a path labelled x from s to an accept state in N_A or N_B .
- Case 1: the path ends in an accept state in N_A . Then there is a path labelled x from s_A to an accept state in N_A . Then x matches A .
- Case 2: the path ends in an accept state in N_B . Then there is a path labelled x from s_B to an accept state in N_B . Then x matches B .
- So x matches A or B , and thus x matches $A \cup B$.

Proof Sketch of Theorem 2

Recursive Step: If A, B are REs...

- $A \cup B$ is a RE
- AB is a RE
- A^* is a RE

4. IS: The RE AB can be matched by the following NFA:

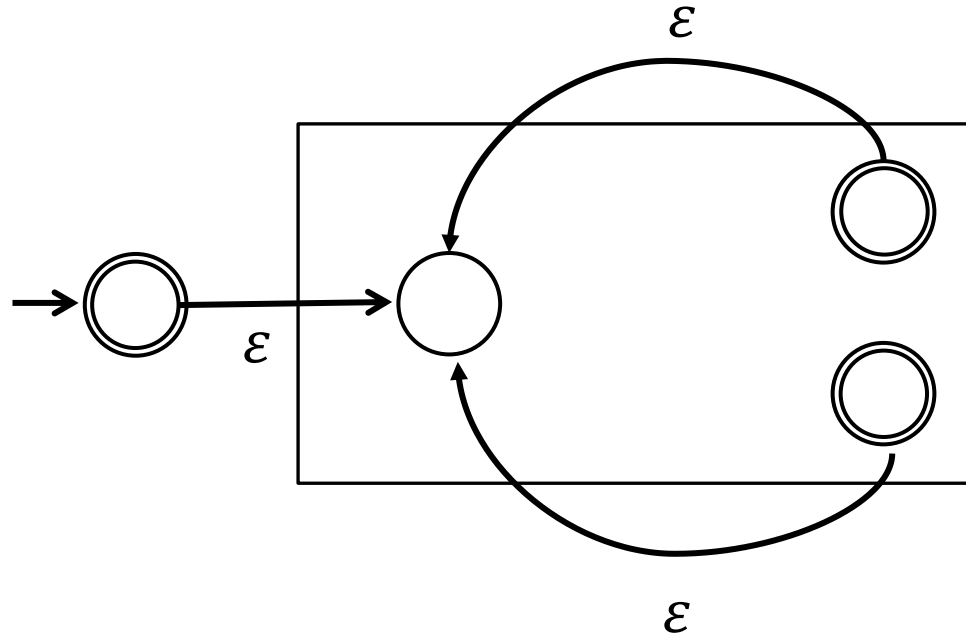


Proof Sketch of Theorem 2

Recursive Step: If A, B are REs...

- $A \cup B$ is a RE
- AB is a RE
- A^* is a RE

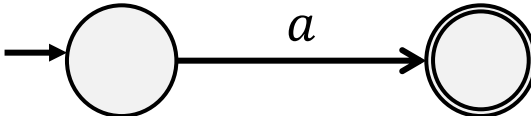
4. IS: The RE A^* can be matched by the following NFA:



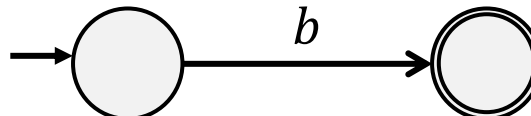
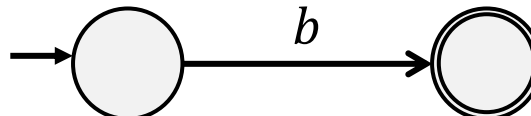
5. Conclusion: (omitted)

Example of RE to NFA Construction

The proof of Theorem 2 gives us a process to write every RE as an NFA. For example, write the RE $(a \cup b)^*c$ as an NFA using the proof construction.

1. To write a , we can use the NFA  → 

The diagram shows a Non-deterministic Finite Automaton (NFA) for the regular expression 'a'. It consists of two states: a start state on the left and an accepting state on the right. The start state is represented by a single circle with an incoming arrow from the left. The accepting state is represented by a double circle. A single transition arrow labeled 'a' connects the start state to the accepting state.

2. To write b , we can use the NFA  → 

The diagram shows a Non-deterministic Finite Automaton (NFA) for the regular expression 'b'. It consists of two states: a start state on the left and an accepting state on the right. The start state is represented by a single circle with an incoming arrow from the left. The accepting state is represented by a double circle. A single transition arrow labeled 'b' connects the start state to the accepting state.

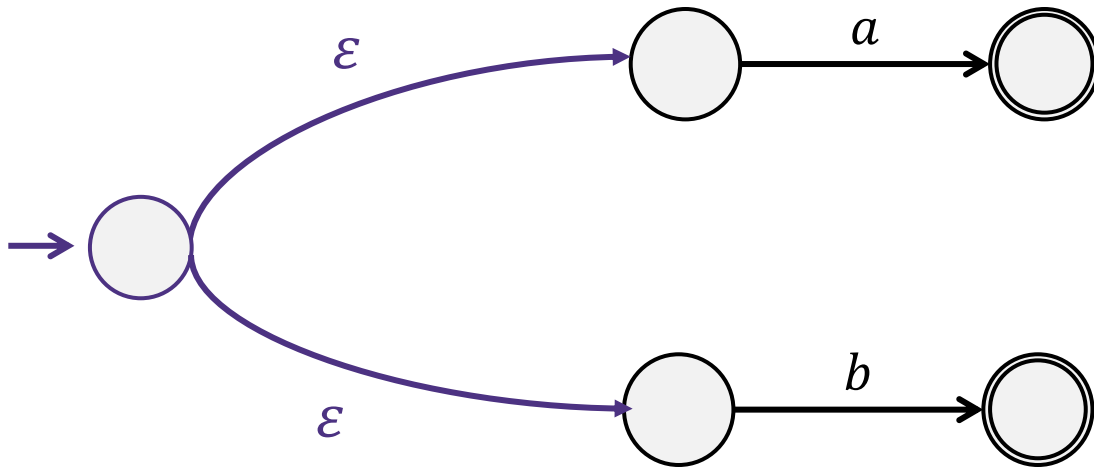
3. To write c , we can use the NFA  → 

The diagram shows a Non-deterministic Finite Automaton (NFA) for the regular expression 'c'. It consists of two states: a start state on the left and an accepting state on the right. The start state is represented by a single circle with an incoming arrow from the left. The accepting state is represented by a double circle. A single transition arrow labeled 'c' connects the start state to the accepting state.

Example of RE to NFA Construction

The proof of Theorem 2 gives us a process to write every RE as an NFA. For example, write the RE $(a \cup b)^*c$ as an NFA using the proof construction.

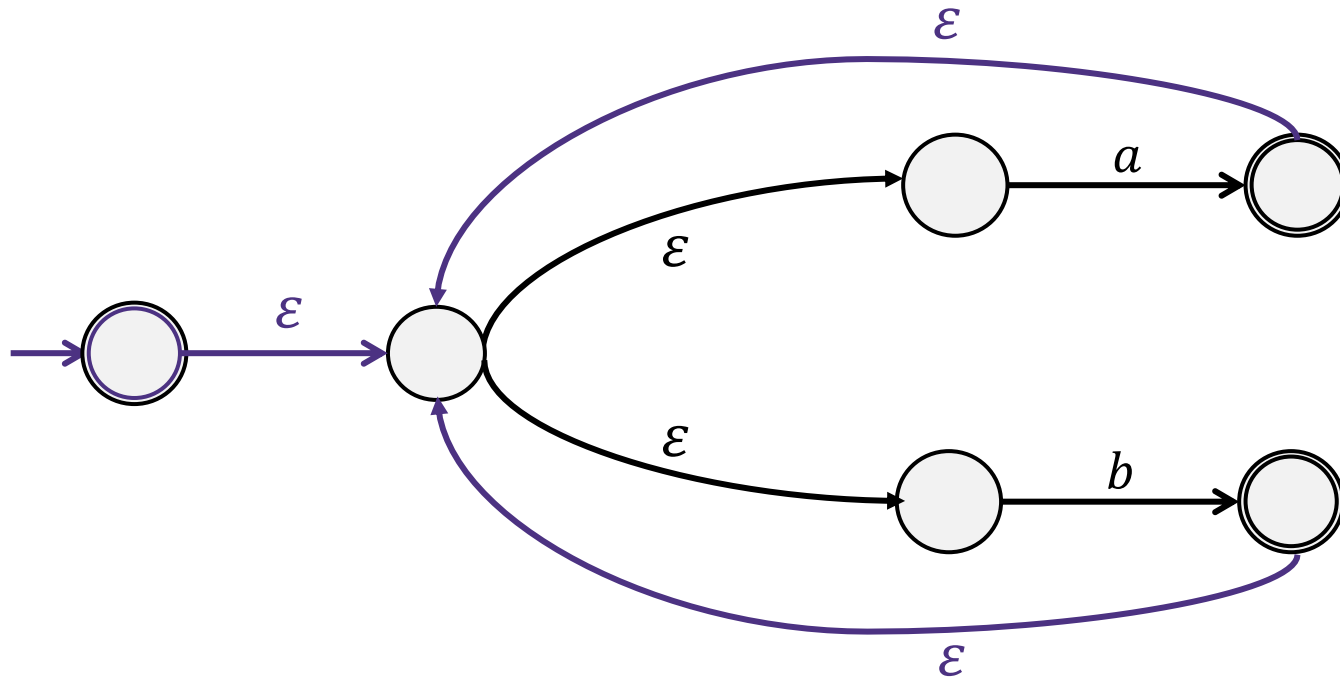
4. Construct the NFA recognizing $a \cup b$.



Example of RE to NFA Construction

The proof of Theorem 2 gives us a process to write every RE as an NFA. For example, write the RE $(a \cup b)^*c$ as an NFA using the proof construction.

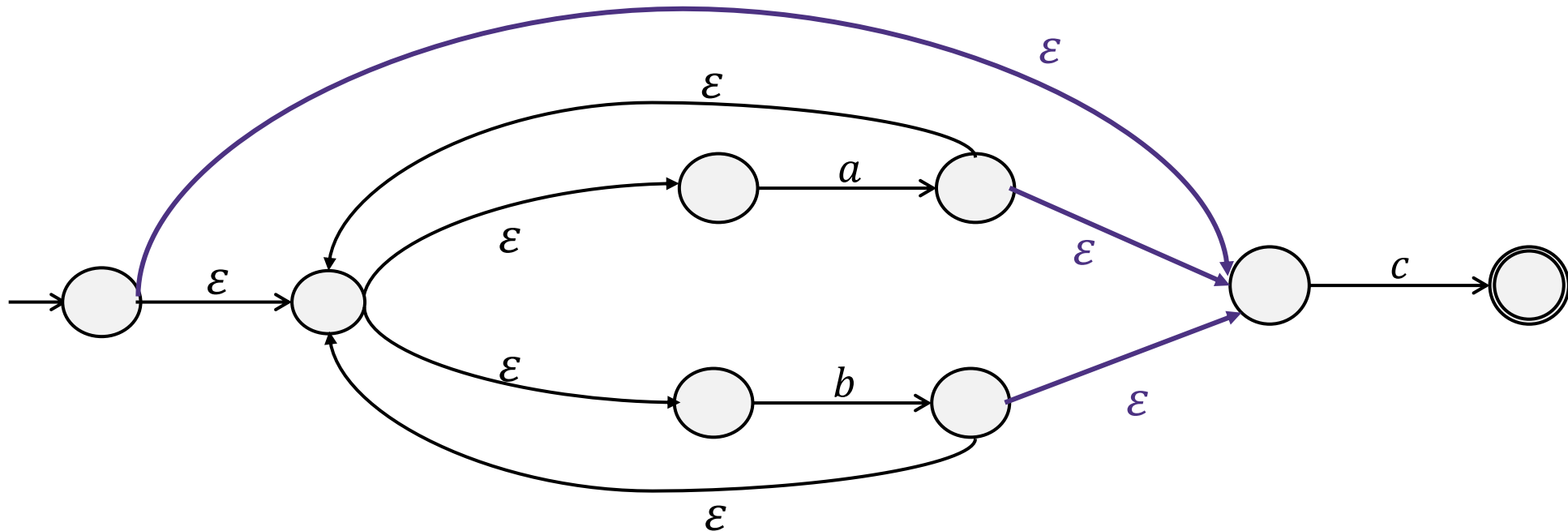
5. Construct the NFA recognizing $(a \cup b)^*$.



Example of RE to NFA Construction

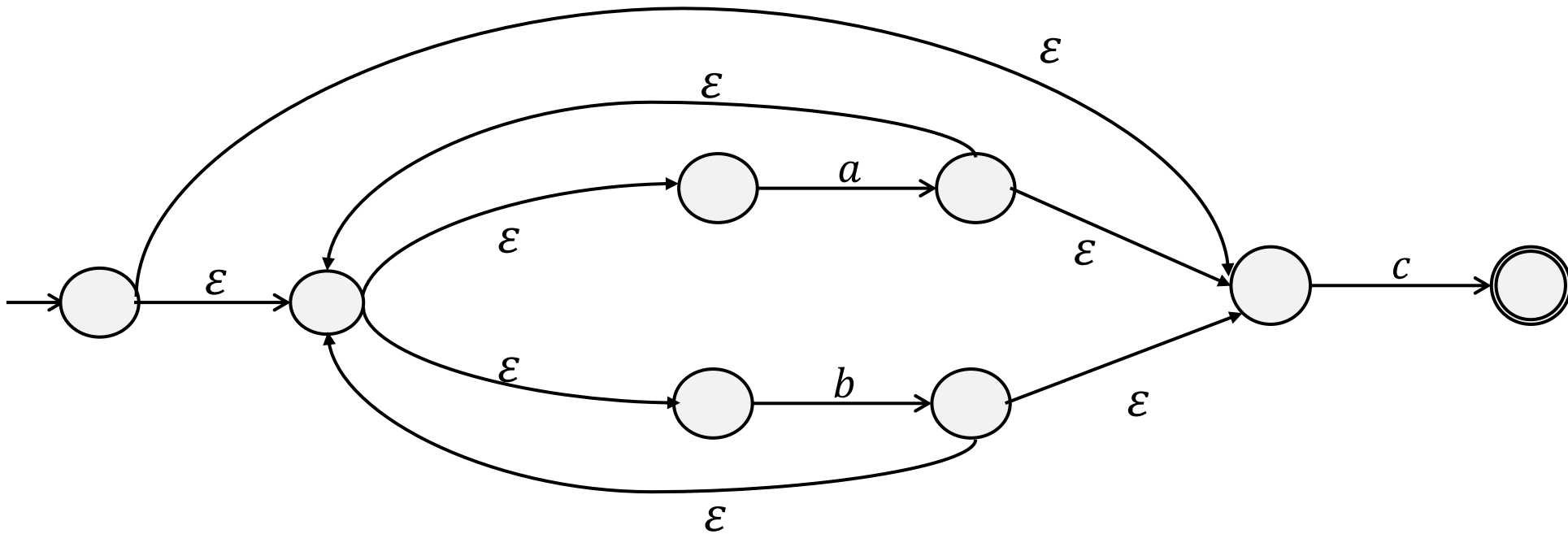
The proof of Theorem 2 gives us a process to write every RE as an NFA. For example, write the RE $(a \cup b)^*c$ as an NFA using the proof construction.

6. Construct the NFA recognizing $(a \cup b)^*c$.



Example of RE to NFA Construction

Final NFA construction for the RE $(a \cup b)^*c$:



Note: This construction will not necessarily give us the simplest NFA.

The story so far...

REs

\supseteq

CFGs

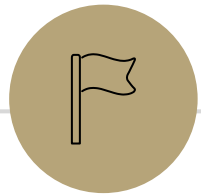
\supseteq

\supseteq

DFAs

$=$

NFAs



NFAs \subseteq REs



The story so far...

REs

\subseteq

CFGs

\subseteq

\subseteq

DFAs

$=$

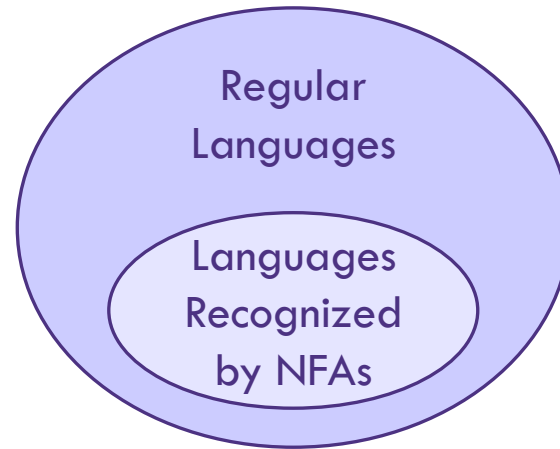
NFAs

Are these \subseteq really " $=$ " or " \subsetneq "?

NFAs and REs

Theorem 3:

For any language A recognized by an NFA, there is a RE that matches A .



We won't cover this construction, but [here](#) are optional notes if you are interested.

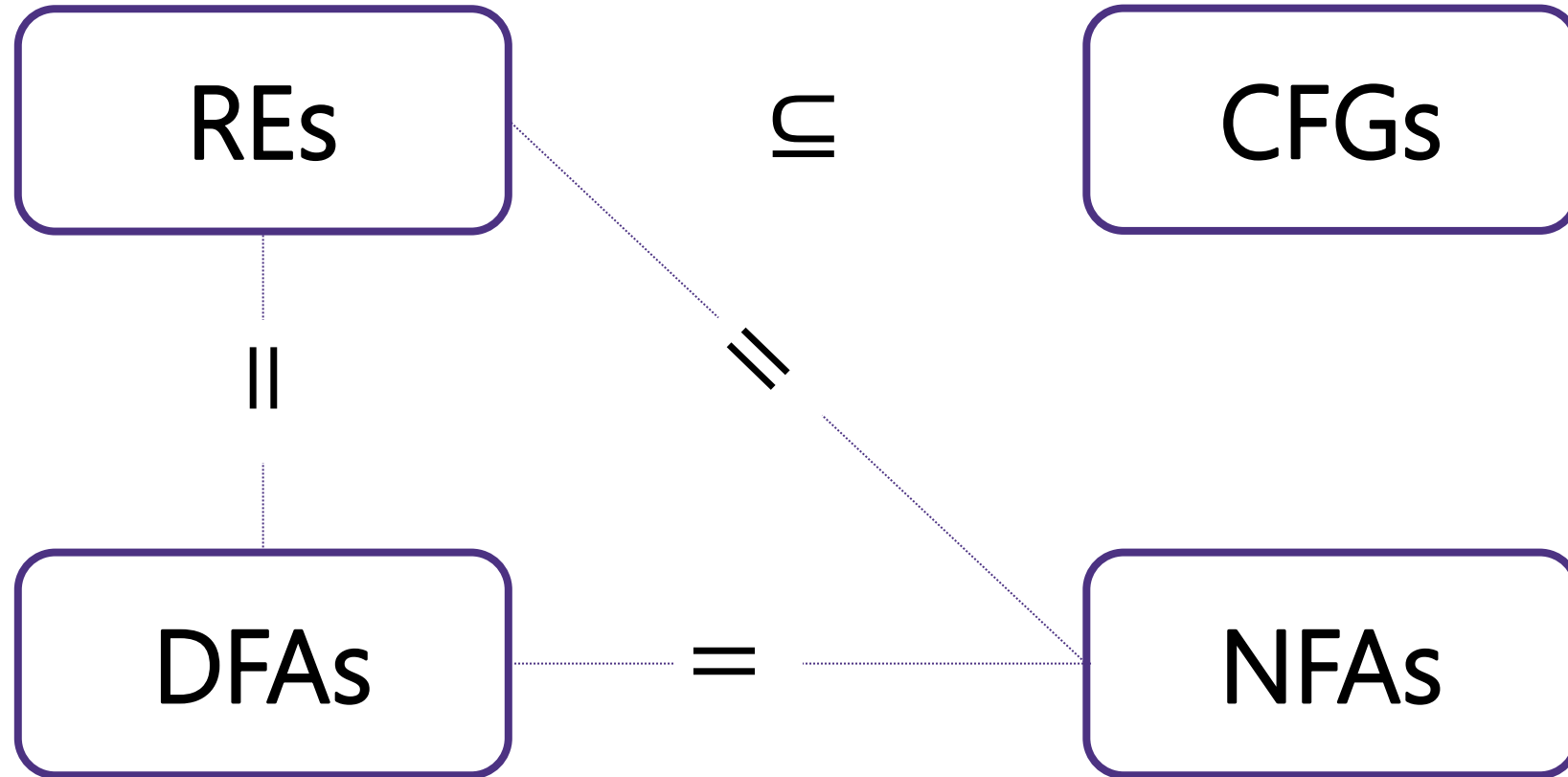
Regular Languages

Corollary:

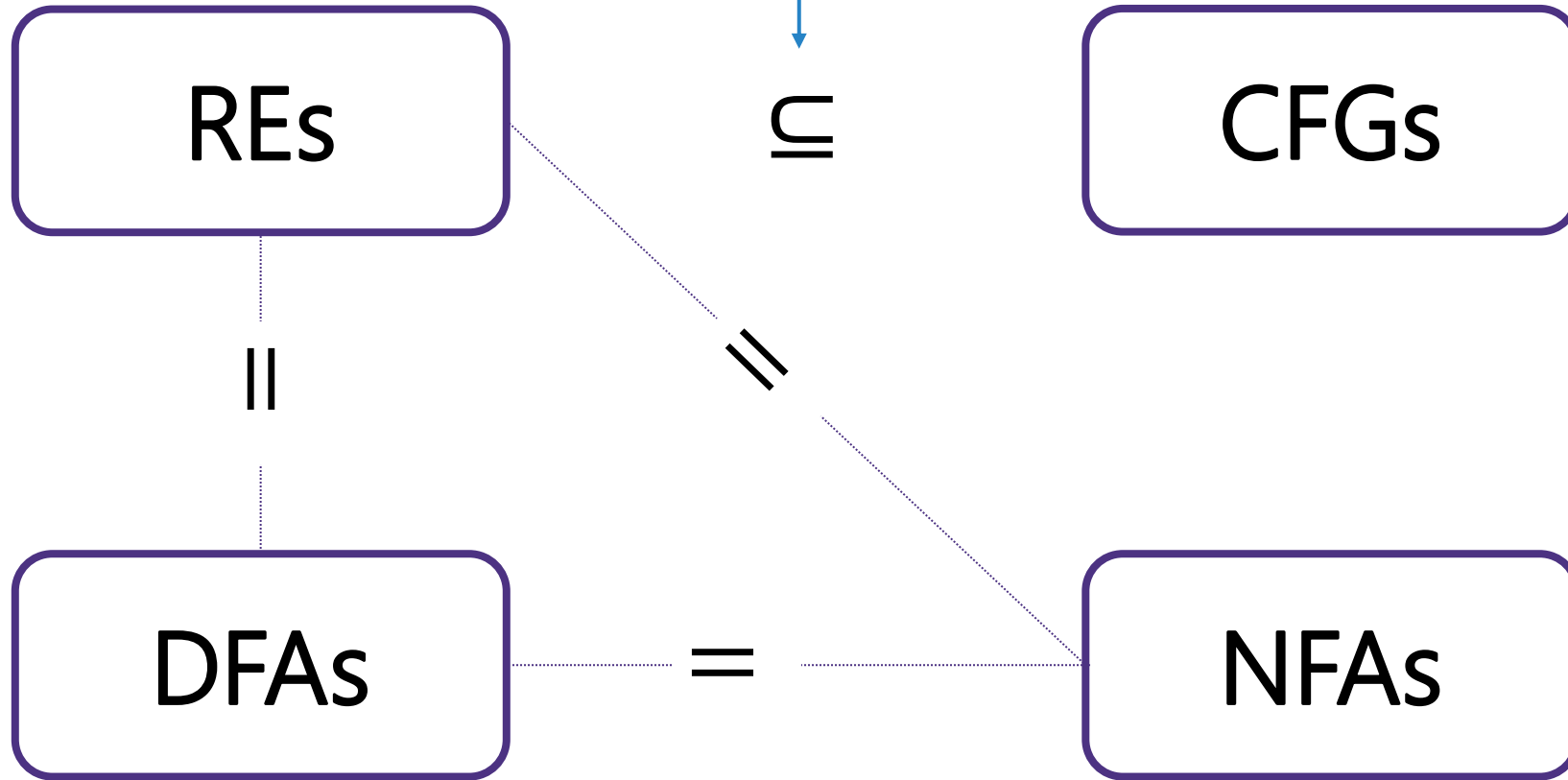
A language is recognized by a DFA iff it is recognized by an NFA, iff it is matched by a RE.



The story so far...



The story so far...



Next time: Proving that this is actually \subsetneq , & in general how to prove a language is irregular

What do I need to know for the Final?

- We won't ask you to replicate any of the constructions.
 - NFA to DFA (Friday's Class)
 - RE to CFG (Theorem 1)
 - RE to NFA (Theorem 2)
 - NFA to RE (Theorem 3)
- We will ask you True / False, Multiple Choice, or short-answer questions that assess your knowledge of the relationships, and the overall proof concepts.
 - E.g. "Is there an irregular language that can be recognized by a DFA?"

Feedback: Course Evaluations

We'd **really** appreciate if you take the time right now to fill out the course evaluations! The link should be in your email.

THANK
YOU!