The standard construction

- We want to show that any regular expression has an \( \text{NFA}_\varepsilon \)
- We do this by always creating \( \text{NFA}_\varepsilon \)'s that have a SINGLE FINAL STATE that is different from \( q_0 \)
- We show how to combine them to build larger regular expressions
- The basic regular expressions are \( \emptyset \), \( \varepsilon \), and \( a \) and the simplest \( \text{NFA}_\varepsilon \)'s with a single initial and final state are:

---

NFA\( _\varepsilon \)'s for the basic regular expressions

\( \emptyset \)

\( \varepsilon \)

\( a \)

---

The NFA\( _\varepsilon \) for a sum of regular expressions

- If we have constructed the NFA\( _\varepsilon \)'s (each with a single final state) for two regular expressions \( r_1 \) and \( r_2 \) then the NFA\( _\varepsilon \) for \( r_1 + r_2 \) is shown next
- The two final states from \( r_1 \) and \( r_2 \) are no longer final. A new single final state is added with \( \varepsilon \)-transitions from the old final state
The NFA$\epsilon$ for a *sum* of regular expressions - II

- What were final states in the graphs of $r_1$ and $r_2$ are no longer final states (they are shown with dotted lines around them).
- There is only one final state.

The NFA$\epsilon$ for a *concatenation* of regular expressions

- If we have constructed the NFA$\epsilon$'s (each with a single final state) for two regular expressions $r_1$ and $r_2$ then the NFA$\epsilon$ for $r_1r_2$ is shown next.
The NFA\(\varepsilon\) for the *Kleene closure* of a reg. expr.

- If we have constructed the NFA\(\varepsilon\) (each with a single final state) for the regular expression \(r\) then the NFA\(\varepsilon\) for \(r^*\) is shown next

Simplifying the construction

- The problem with the \(r^*\) construction is that it generates a lot of states and a lot of \(\varepsilon\)-transitions
- Two simplifications are often *but not always* possible

No transitions into the initial state

- If there are no transitions *into* the initial state in the original graph for \(r\), we can make the simplification above

No transitions out of the final state

- If there are no transitions *out of* the final state in the original graph for \(r\), we can make the simplification above
Combination of the simplifications

- If there are no transitions into the initial state or out of the final state, then the simplified picture is:

Simplification is not always possible

- We have examples to show that you cannot always make these simplifications, first a simple case where everything works OK
  - $a \rightarrow \varepsilon \rightarrow a$
  - $a^* \rightarrow \varepsilon \rightarrow a$

But it can be simpler

- Simpler $a^*$

Case where only partial simplification is impossible

- $a(ba)^*$
  - $b \rightarrow a$

- $(a(ba)^*)^*$
  - $b, \varepsilon \rightarrow a \rightarrow \varepsilon \rightarrow \varepsilon \rightarrow \varepsilon$
Other automata are possible

- alternative for \((a(ba)^*)\)

Final state simplification fails

- Merging the final states in the figure on slide 3-62 gives an NFA for 
  \(\varepsilon + (\varepsilon + b)(a + ab)^*a\)
- This language includes \(ba\), which is wrong

Initial state simplification fails

- Merging the initial states in the figure on slide 3-62 produces an NFA for 
  \((a + ab)^*(\varepsilon + a)\) (includes \(ab\))

Merging on both sides also fails

- The effect of merging initial states and final states: 
  \((a + b + ab)^*(\varepsilon + a)\)
### Other examples

- Some on-line notes explore an NFA\(\epsilon\) for \((ab)^*\)^*, where merging the final states changes the language accepted (notes3.html).
- Regular expressions are used in several tools in UNIX. Some of the on-line notes give examples. (notes2.html), e.g.
- pp. 124-7 -> pp. 124-27

### Pumping Lemma

- We postpone showing how to convert an NFA\(\epsilon\) to a regular expression and how to minimize an automaton.
- Instead we show how to prove a language is not regular.
- We need the pumping lemma for regular languages (PL4RL).
- The PL4RL is *not* used to prove languages *are* regular.

### Statement of theorem

The PL4RL states that if \(L\) is a regular language, then there is an integer \(n\) such that when we take any \(z \in L\) with \(|z| \geq n\) (if any such \(z\) exist), we must be able to write \(z = uvw\), where
- \(|uv| \leq n\)
- \(|v| > 0\)
- \(uv^k w \in L\) for all \(k \geq 0\).
- In particular, if one such \(z\) exists, \(L\) is infinite.

### More general PL4RL

- The PL4RL can be understood with a few graphics, which will be our proof.
- Also we might as well prove a more general form:
More general PL4RL

- For any regular language $L$ there is an $n$ (for that $L$) such that if $z = uw_1w_2...w_nv \in L$, where $u$, $w_1$, $w_2$, ..., $w_n$, $v$ are all strings, then there is a loop among the $w$’s and $uw_1w_2...(w_{i+1}...w_j)^k...w_nv \in L$ for all $k \geq 0$

The string $z$

- If $z = uw_1w_2...w_nv \in L$, where $n$ is the number of states in a DFA for $L$, then there is a path in the DFA from $q_0$ to a final state:

How many $p_i$?

- Now, even if $u = \varepsilon$ and $v = \varepsilon$, in which case $q_0 = p_0$ and $p_n$ is final, there are $n + 1$ states from $p_0$ to $p_n$ inclusive

Pigeon-hole principle

- BUT the DFA only has $n$ states so two of the $p$’s are the same
- Suppose $p_i = p_j$ for some $i$, $j$, with $0 \leq i < j \leq n$
The loop - I

- Concentrate on the loop
- The path shown from \( q_0 \) to a final state, continues to reach the final state independent of the number of times the loop is traversed

Skip the loop

Repeat the loop

- Loop omitted
- Loop repeated several times
The string can be pumped

- The string \( uw_1w_2... (w_{i+1}... w_j)^k... w_n v \)
describes a path from the initial state to a final state for all \( k \geq 0 \)
- Hence all these strings are in \( L \)
- The version of the PL4RL announced at the beginning follows from the
general result by setting \( z = a_1a_2...a_nx \), where \( a_1, a_2, ..., a_n \) are the
first \( n \) symbols in the string \( z \)

Example

- Consider \( L = \{ww^R : w \in (0 + 1)^*\} \),
where \( w^R \) denotes the string \( w \) written in reverse
- \( L \) is the set of even-length palindromes

Applying the PL4RL

- If \( L \) is regular there is an \( n \) as in the PL4RL
- Pick \( z = (01)^n(10)^n = w_1w_2...w_nv \), so \( u = \varepsilon \), \( w_1 = w_2 = ... = w_n = 01 \), \( v = (10)^n \)
- Then there are \( i \) and \( j \) so that
\( w_1w_2...(w_{i+1}... w_j)^k... w_n v \) is in \( L \) for all \( k \)
- Now, \( w_1w_2...(w_{i+1}... w_j)^0... w_n v = w_1w_2...w_iw_{i+1}... w_n v = (01)^{n-j+i}(10)^n \)
- The mid-point of string needs 00 or 11 but that pattern is not at center

Example 1

- Consider \( L = \{0^a1^{2b} : 3a > 2b\} \) where \( a, b \) are some integers.
- Prove that this is not regular.
- We Prove this by contradiction, using PL4RL
### Details of the exponents and conditions

- The notation $1^{2b}$ indicates that there is an even number of 1’s.
- $3a$ has to be greater than $b$ (3 times number of 0’s has to be greater than number of 1’s).

### Applying the PL4RL

- If $L$ is regular there is an $n$ as in the PL4RL.
- Pick $z = 0^{2n+1}1^n = uvw$, where $|uv| < n$, $|v| > 1$ and
- $uv^kw$ should belong to $L$ for all $k$ greater than or equal to zero.

### Pump down

- Now we have $z = 0000...00011111111...111$

  $2n + 1 \quad 6n$

- So $uv$ contains only 0’s (since $|uv| < n$)
- $v$ contains only 0’s
- $uv^0w = uw = 0^{2n+1-|v|}1^n$
- Since, $|v| \geq 1$, $3(2n + 1 - |v|) \leq 6n$

### Conclusion

- Hence $uv^kw \notin L$ for $k = 0$
- Thus, $L$ violates the PL4RL
- So $L$ cannot be regular
Example 2

- Consider $L = \{ 0^p : p \text{ prime} \}$.
- Prove that this is not regular.
- *We Prove this by contradiction, using PL4RL*

---

About prime numbers

- Theorem: There are an infinite number of primes. (this means, given a number, we can always find a prime number greater than it).
- We can prove this easily by contradiction...
  - if $p_1, p_2, \ldots, p_k$ was the finite sequence of all primes, then $p_1p_2\ldots p_k+1$ is a new prime (none of the other $p_j$’s is a factor of this new number) -- contradiction

---

Applying the PL4RL

- If $L$ is regular there is an $n$ as in the PL4RL
- Take any prime $p > n$
- Pick $z = 0^p = uvw$
  - where $|uv| < n$, $|v| > 1$ and
  - $uv^kw$ should belong to $L$ for all $k$ greater than or equal to zero.

---

Pump up

- Now in $uv^kw$, take $k = p+1$
- $z = uvvvvv\ldots vvvvww$
  - $p+1$ copies of $v$
- $z = 0^{p+p/|v|}$ since we added $p$ extra $v$’s
  - $= 0^{p(1+|v|)}$
Pump up

- Is \( p(1+|v|) \) a prime number ???

- No, \( (1+|v|) \geq 2 \)
  
  so, \( p(1+|v|) \) is a composite number

- Hence \( uv^k w \notin L \) for \( k = p + 1 \)

- Thus, \( L \) violates the PL4RL. So \( L \) cannot be regular

Example 3

- Consider \( L = \{ 0^a1^b2^c : 2a < c, b > 0 \} \).

- Prove that this is not regular

  - We prove this by contradiction, using PL4RL

Details of the exponents and conditions

- \( 2a \) has to be less than \( c \) (2 times number of 0’s has to be less than number of 2’s).

- There has to be one or more 1’s \((b>0)\).

Applying the PL4RL

- If \( L \) is regular there is an \( n \) as in the PL4RL

  - Pick \( z = 0^n1^22^{(n+1)} = uvw \), where \( |uv| < n, |v| \geq 1 \) and

  - \( uv^kw \) should belong to \( L \) for all \( k \) greater than or equal to zero
Applying PL4RL

- $z = 000\ldots000111\ldots111122222\ldots22$

  \[n \quad n \quad 2n+2\]

- Since $|uv| \leq n$, so from above $uv$ contains only 0’s
- Therefore, $v$ contains only 0’s

Pump up

- $uv^3w = 0^{n+2}v^12^{2n+2}$
- Let’s see if it belongs to $L$, that is, if it satisfies the condition $2(n + 2|v|) < 2n + 2$
- Since $|v| \geq 1$, the above is clearly false
- Hence $uv^kw \notin L$ for $k = 3$
- Thus, $L$ violates the PL4RL. So $L$ cannot be regular