

1.3 Regular Expressions

- These have an important role in describing patterns in searching for strings in many applications (e.g. awk, grep, Perl, ...)

All *regular expressions* of alphabet Σ are

1. \emptyset and ε are regular expressions,
2. a is a regular expression of Σ for all $a \in \Sigma$,
3. if R_1 and R_2 are regular expressions, then also
 - $(R_1 \cup R_2)$,
 - $(R_1 \circ R_2)$ and
 - R_1^*
 are regular expressions



Each regular expression R of Σ represents a language $L(R)$

1. $L(\emptyset) = \emptyset$,
2. $L(\varepsilon) = \{\varepsilon\}$,
3. $L(a) = \{a\} \quad \forall a \in \Sigma$,
4. $L((R_1 \cup R_2)) = L(R_1) \cup L(R_2)$,
5. $L((R_1 \circ R_2)) = L(R_1) \circ L(R_2)$ and
6. $L(R_1^*) = (L(R_1))^*$

Proper closure: R^+ is a shorthand for RR^* (Kleene plus)

Observe: $R^+ \cup \varepsilon = R^*$

- Let R^k be shorthand for the concatenation of k R 's with each other.



Examples

$0^*10^* = \{ w \mid w \text{ contains a single } 1 \}$

$\Sigma^*001\Sigma^* = \{ w \mid w \text{ contains the string } 001 \text{ as a substring} \}$

$1^*(01^+)^* = \{ w \mid \text{every } 0 \text{ in } w \text{ is followed by at least one } 1 \}$

$(\Sigma\Sigma)^* = \{ w \mid w \text{ is a string of even length} \}$

$01 \cup 10 = \{ 01, 10 \}$

$0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{ w \mid w \text{ starts and ends with the same symbol} \}$

$(0 \cup \varepsilon)1^* = 01^* \cup 1^*$

$(0 \cup \varepsilon)(1 \cup \varepsilon) = \{ \varepsilon, 0, 1, 01 \}$

$1^*\emptyset = \emptyset$

$\emptyset^* = \{ \varepsilon \}$



- For any regular expression R

- $R \cup \emptyset = R$ and

- $R \circ \varepsilon = R$

- However, it may hold that

- $R \cup \varepsilon \neq R$ ja

- $R \circ \emptyset \neq R$

For example, the unsigned real numbers that can be recognized using the previous automaton can be expressed with the regular expression

$$d^+(\cdot d^+ \cup \varepsilon)(E(+ \cup - \cup \varepsilon) d^+ \cup \varepsilon),$$

where $d = (0 \cup \dots \cup 9)$



Theorem 1.54 *A language is regular if and only if some regular expression describes it.*

We state and prove each direction of this theorem separately.

Lemma 1.55 *If a language is described by a regular expression, then it is regular.*

Proof. Any regular expression can be converted into a finite automaton, which recognizes the same language as that described by the regular expression.

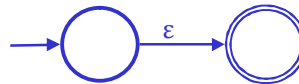
There are only six rules by which regular expressions can be composed. The following pictures illustrate the NFA for each of these cases. \square



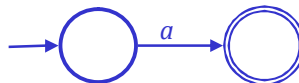
$r = \emptyset$

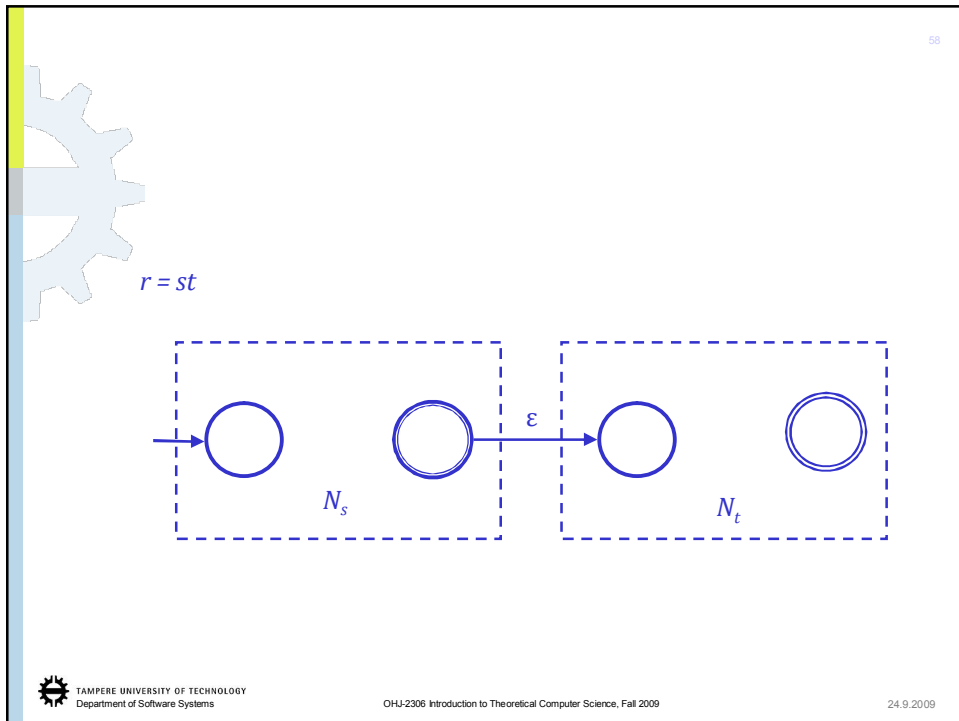
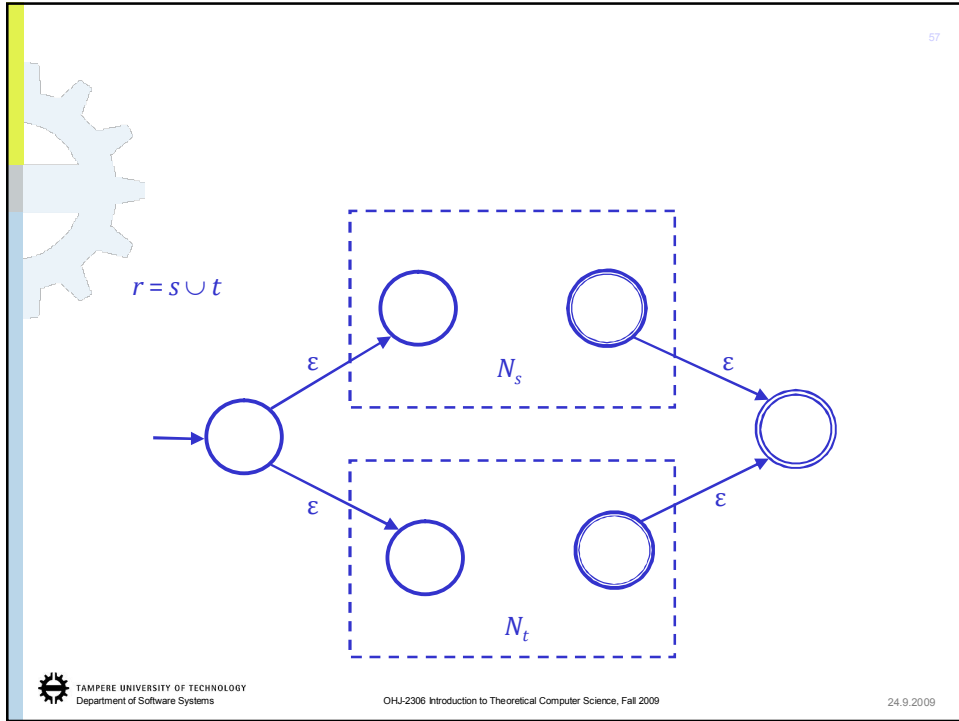


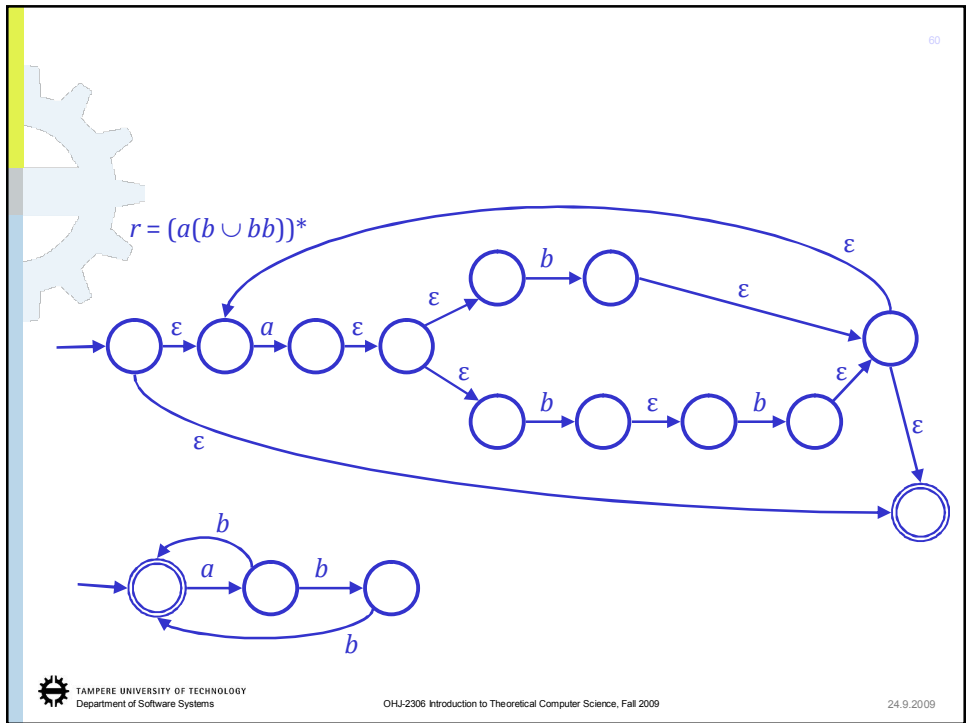
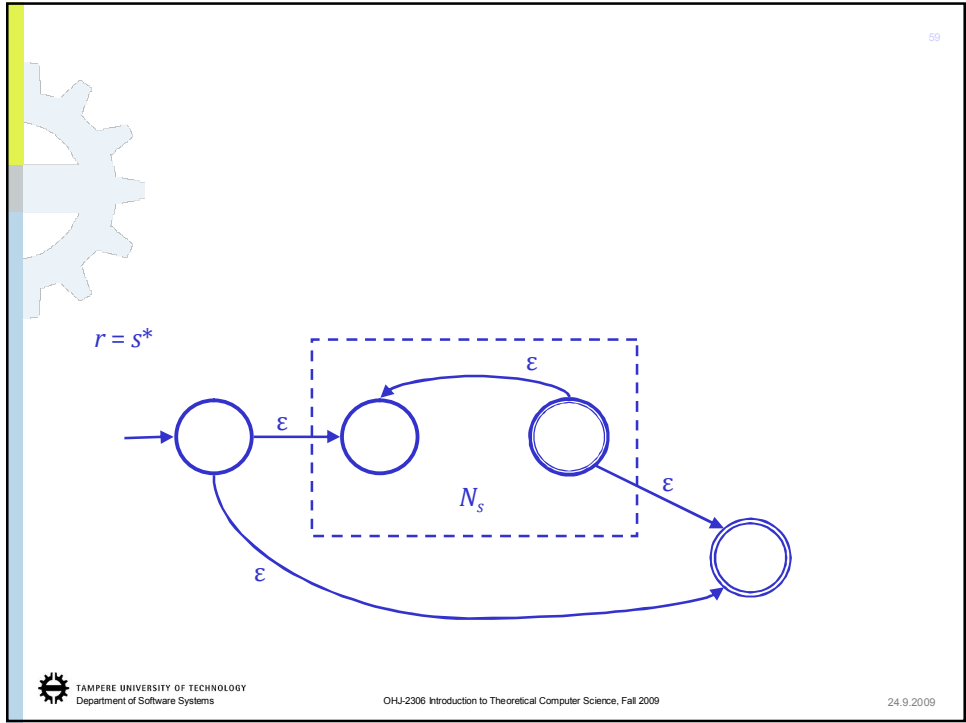
$r = \varepsilon$



$r = a$







Lemma 1.60 *If a language is regular, then it is described by a regular expression.*

Proof. By definition a regular language can be recognized with a (nondeterministic) finite automaton, which can be converted into a generalized nondeterministic finite automaton (GNFA). The GNFA finally yields a regular expression that is equivalent with the original automaton.

Let RE_{Σ} denote the set of regular expressions over Σ

- In a GNFA the transition function δ is a *finite mapping*

$$\delta: Q \times RE_{\Sigma} \rightarrow \mathcal{P}(Q)$$

- $(q, w) > (q', w')$ if $q' \in \delta(q, r)$ for some $r \in RE_{\Sigma}$ s.t. $w = zw', z \in L(r)$



A GNFA M can be reduced into a regular expression which describes the language recognized by M

1. We compress M into a GNFA with only 2 states (so that the language recognized remains equivalent)

1. The accept states of M are replaced by a single one (ϵ arrows)

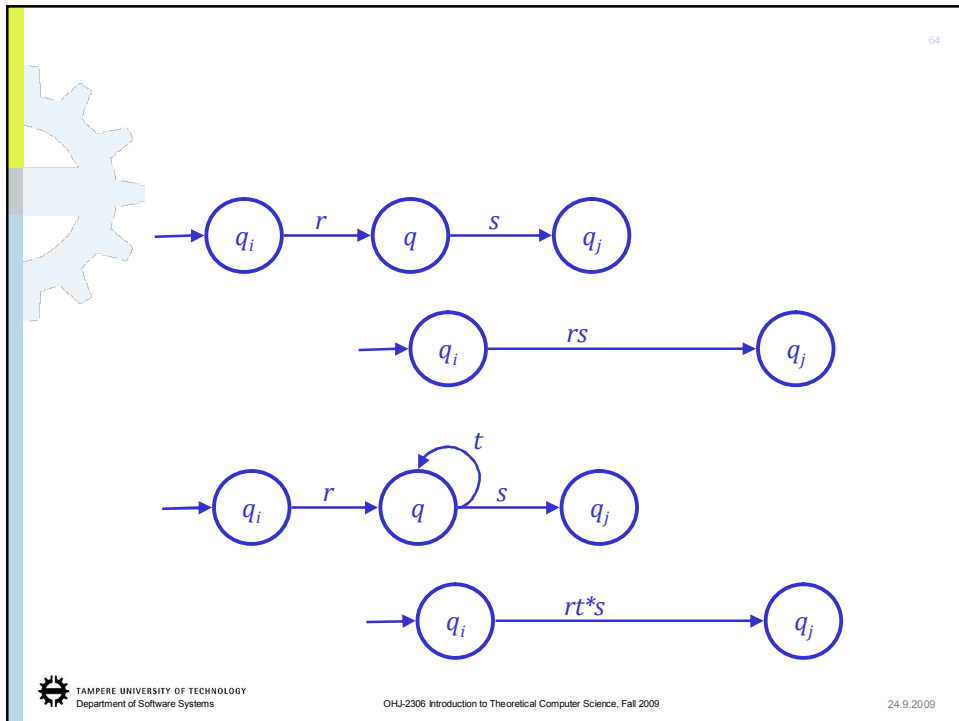
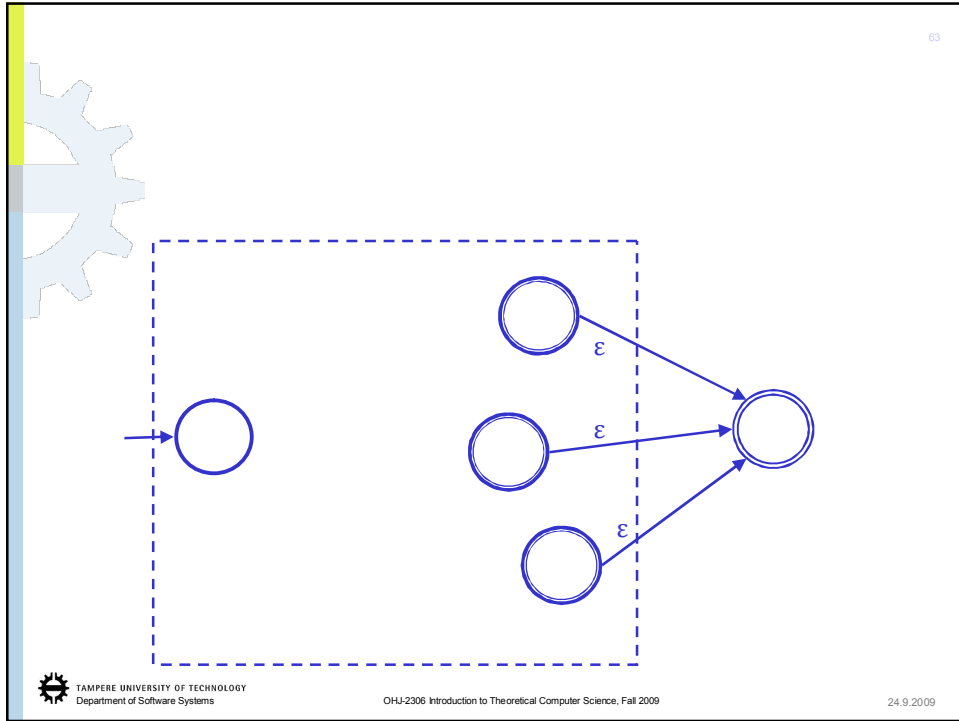
2. We remove all other states q except the start state and final state.

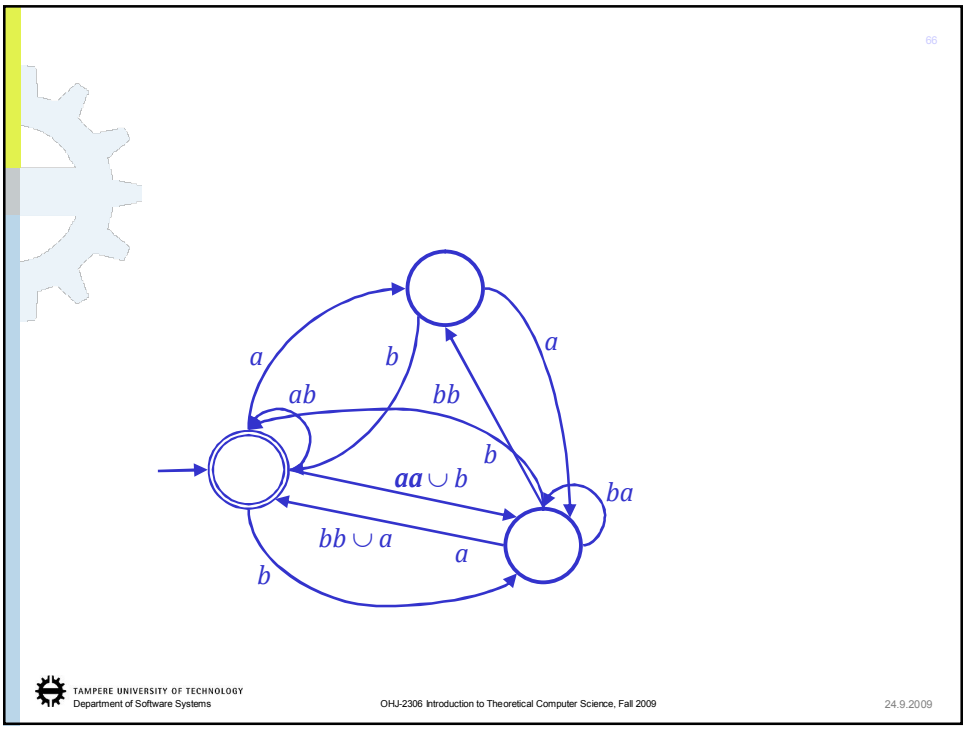
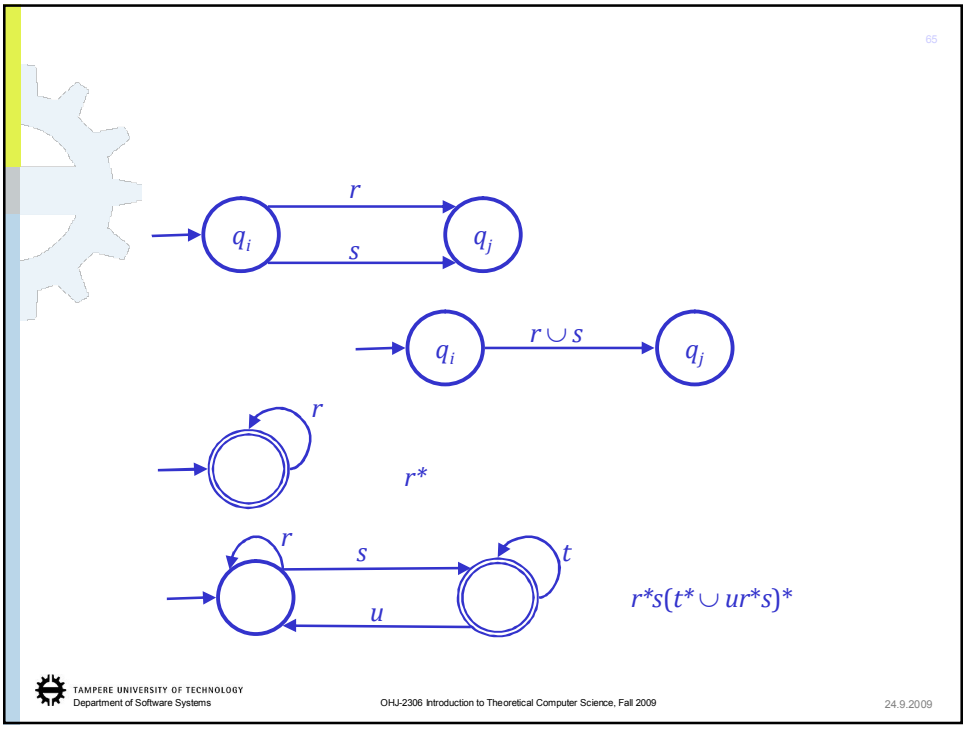
Let q_i and q_j be the predecessor and successor of q on some route passing through q .

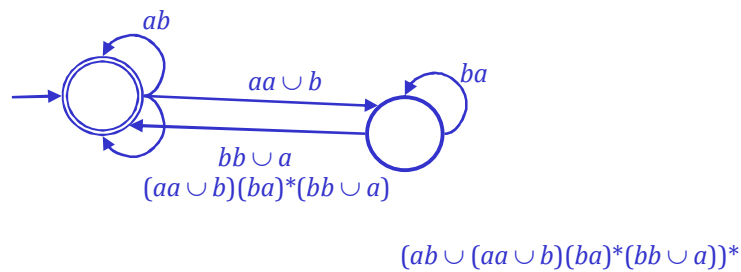
Now we can remove q and rename the arrow between q_i and q_j with a new expression.

2. Eventually the GNFA contains at most two states. It is easy to convert the language recognized into a regular expression.









1.4 Nonregular Languages

- The number of formal languages over any alphabet (= decision/recognition problems) is uncountable
- On the other hand, the number of regular expressions (= strings) is countable
- Hence, all languages cannot be regular
- Can we find an intuitive example of a nonregular language?
- The language of balanced pairs of parentheses

$$L_{\text{parenth}} = \{ ({}^k \mid k \geq 0 \}$$



Theorem 1.70 (Pumping lemma)

Let A be a regular language. Then there exists $p \geq 1$ (the pumping length) s.t. any string $s \in A$, $|s| \geq p$, may be divided into three pieces, $s = xyz$, satisfying the following conditions:

- $|xy| \leq p$,
- $|y| \geq 1$ and
- $xy^iz \in A \quad \forall i = 0, 1, 2, \dots$

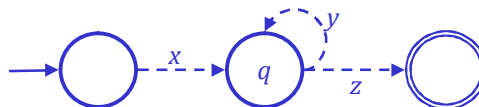
Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that recognizes A s.t. $|Q| = p$. When the DFA is computing with input $s \in A$, $|s| \geq p$, it must pass through some state at least twice when processing the first p characters of s . Let q be the first such state.



Let us choose so that:

- x is the prefix of s that has been processed when M enters q for the first time,
- y is that part of the suffix s that gets processed by M before it re-enters state q , and
- z is the rest of the string s .

Obviously $|xy| \leq p$, $|y| \geq 1$ and $xy^iz \in A$ for all $i = 0, 1, 2, \dots$



□

Observe: The pumping lemma does not give us liberty to choose x and y as we please.



Example

Let us assume that L_{parenth} is a regular language.

By the pumping lemma there exists some number p s.t. strings of L_{parenth} of length at least p can be pumped. Let us choose $s = (p)^p$. Then $|s| = 2p > p$.

By Lemma 1.70 s can be divided into three parts $s = xyz$

s.t. $|xy| \leq p$ and $|y| \geq 1$. Therefore, it must be that

- $x = (i \quad i \leq p-1,$
- $y = (j \quad j \geq 1,$ and
- $z = (p-(i+j))^p.$

By our assumption $xy^kz \in L_{\text{parenth}}$ for all $k = 0, 1, 2, \dots$, but for example

$$xy^0z = xz = (i \quad (p-(i+j))^p = (p-j)^p \notin L_{\text{parenth}},$$

because $p-j \neq p$ since $j \geq 1$.

Hence, L_{parenth} cannot be a regular language

