# CS311 Formal Language and Automata

Tingting Chen

Computer Science

Cal Poly Pomona

# Regular Expressions and Regular Languages

Let $M = (Q, \Sigma, q_0, \delta, A)$ be an FA.

- A string $x \in \Sigma^*$ is accepted by M if $\delta^*(q_0, x) \in A$
- The language accepted (or recognized) by M is the set $L(M) = \{x \in \Sigma^* \mid x \text{ is accepted by } M\}$
- A language L over the alphabet $\Sigma$ is regular iff there is a Finite Automaton that accepts L.

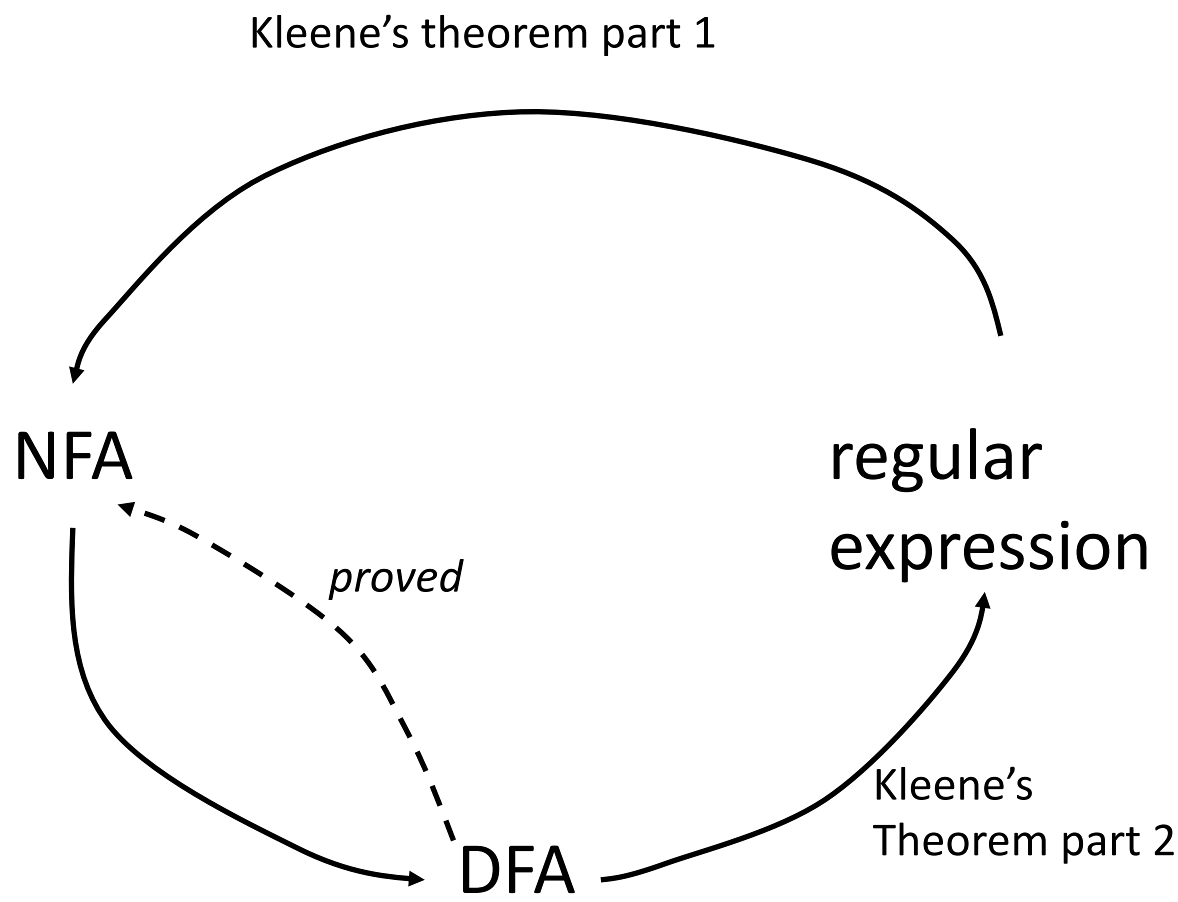
# Kleene’s theorem

1) For any regular expression r that represents language L(r), there is a finite automaton that accepts that same language.

2) For any finite automaton M that accepts language L(M), there is a regular expression that represents the same language.

Therefore, the class of languages that can be represented by regular expressions is equivalent to the class of languages accepted by finite automata -- the *regular languages*.

# Kleene's theorem



# Kleene's theorem – $1^{st}$ half

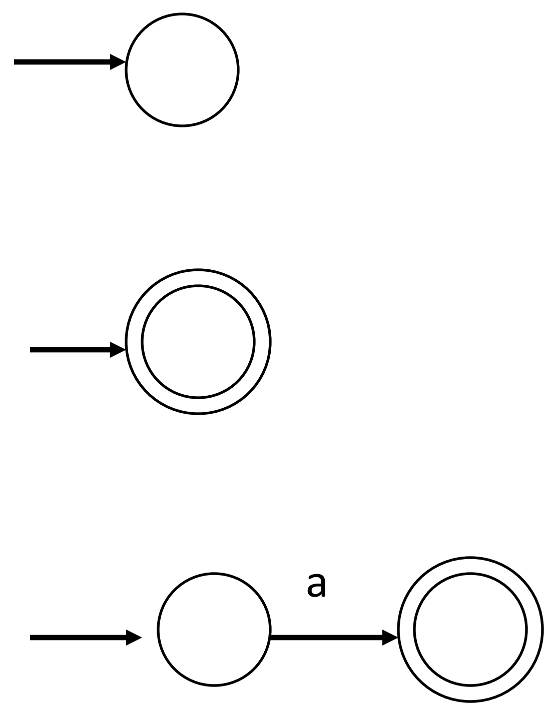
1st half of Kleene's theorem: Let r be a regular expression.  Then there exists some nondeterministic regular accepter that accepts L(r).  Consequently, L(r) is a regular language.

Proof strategy: for any regular expression,
we show how to construct an equivalent NFA.

Because regular expressions are defined recursively,
the proof is by induction.

# Proof of Kleene's theorem – 1st half

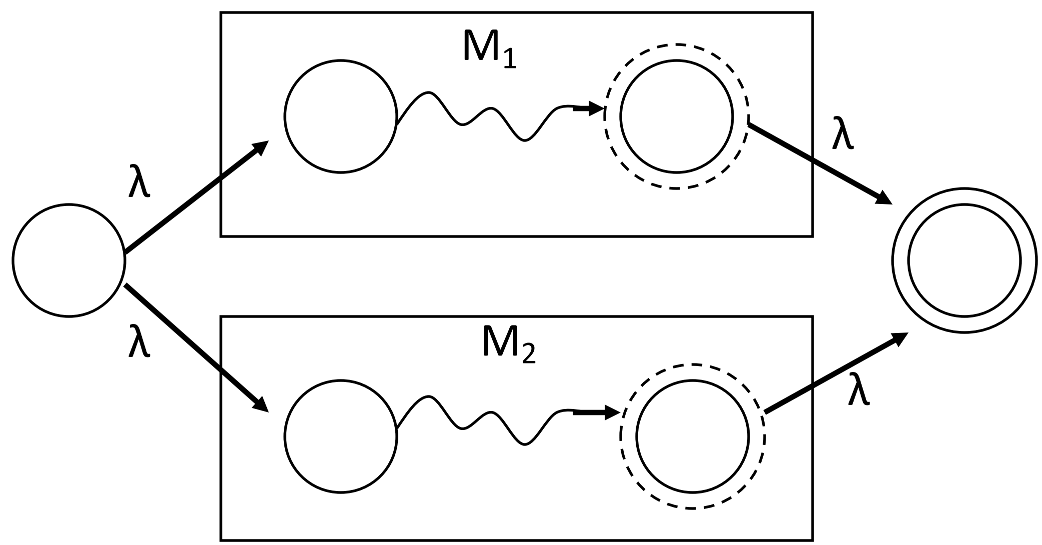
Base step: construct a NFA that accepts each of the simple or "base" languages, $\emptyset$, $\{\lambda\}$, and $\{a\}$ for each $a \in \Sigma$.



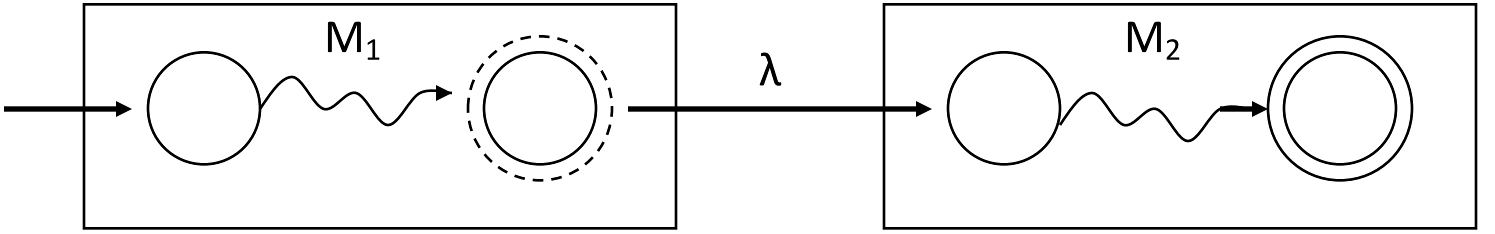
# Proof of Kleene's theorem – $1^{st}$ half

Inductive step:
Assume for regular expressions $r_1$ and $r_2$ that $L(r_1)$ and $L(r_2)$ are accepted by some NFAs, for each of the operations – union, concatenation and star (since "()" is trivial), we will show how to construct an accepting NFA.

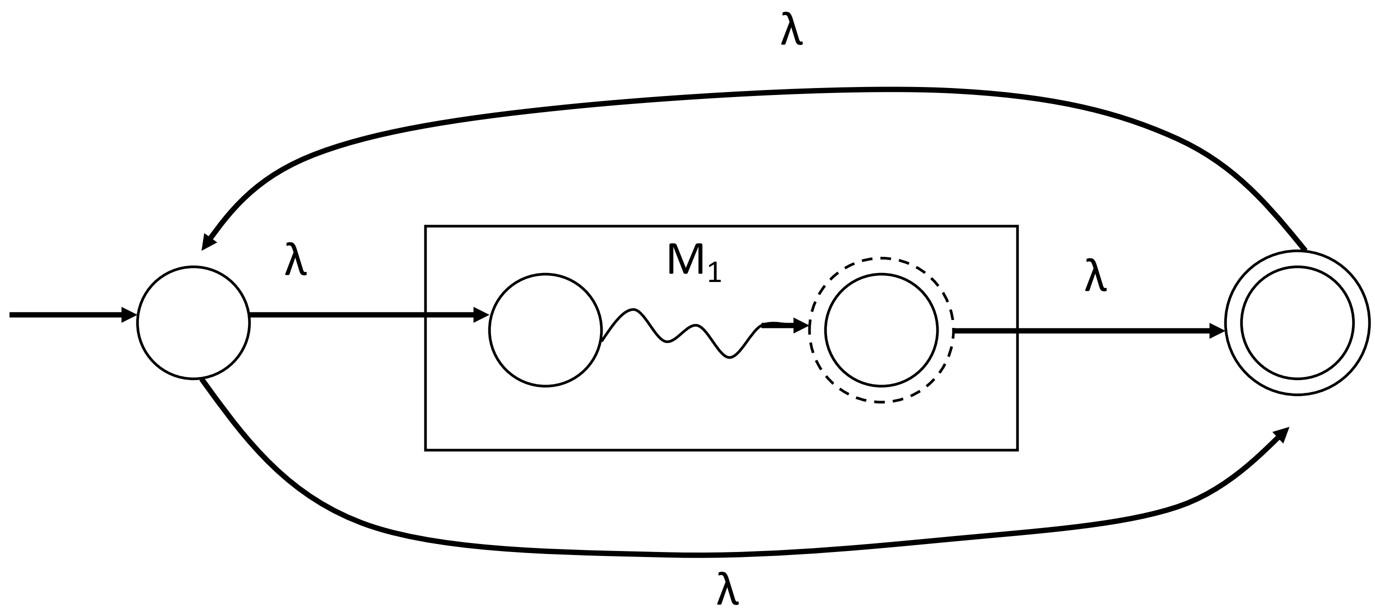
union:



# concatenation:



Star:



# Closure properties of Regular Languages

- Union, concatenation, and Kleene star of two regular languages will result in a regular language, since we can write a regular expression for them.
- Intersection and difference (complement) of two regular languages will also produce a regular language.
- The class of regular languages is said to be **closed** under these operations.

# Kleene’s theorem – 2$^{nd}$ half

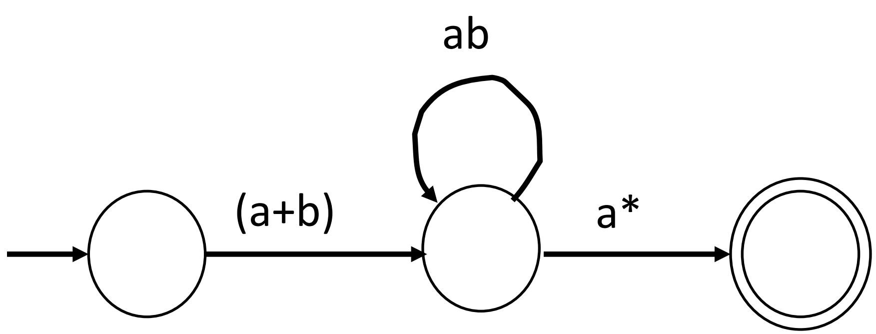
**Kleene’s theorem part 2:** Let L be a regular language. Then there exists a regular expression r such that L = L(r).

Any language accepted by a finite automaton can be represented by a regular expression.

The proof strategy: For any DFA, we show how create an equivalent regular expression. In other words, we describe an algorithm for converting any DFA to a regular expression.

# Generalized Transition Graph

- A generalized transition graph (GTG) is a transition graph whose edges are labeled with regular expressions; otherwise it is the same as the usual transition graph



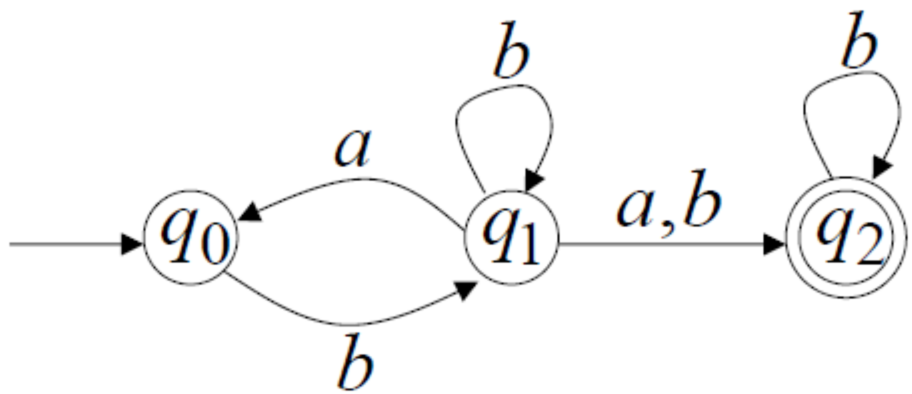
# Converting a DFA into an equivalent regular expression – Informal steps

Initial step: Change every transition labeled a,b to (a+b). Make sure there is a single final state, distinct from its initial state, if needed adding a single final state with incoming λ-transitions from every previous final state.

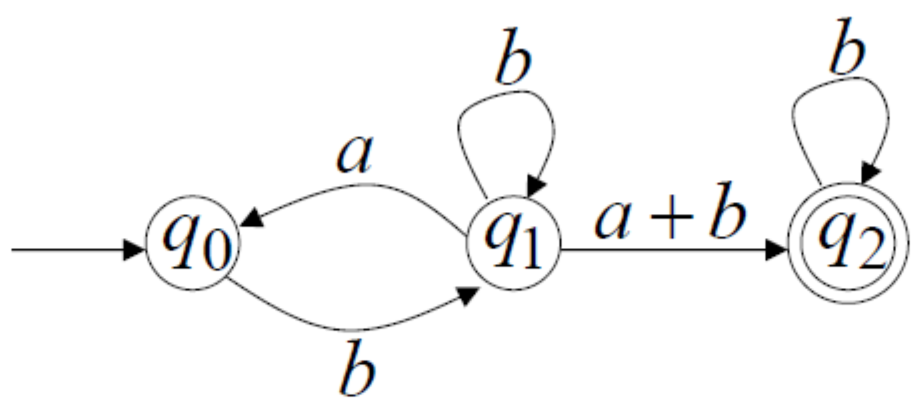
Main step: Until GTG has only two states (initial state and final state), repeat the following:

- -- pick some non-start, non-final state
- -- remove it from the graph and re-label transitions with regular expressions so that the same language is accepted

# Converting a FA into an equivalent regular expression – Example

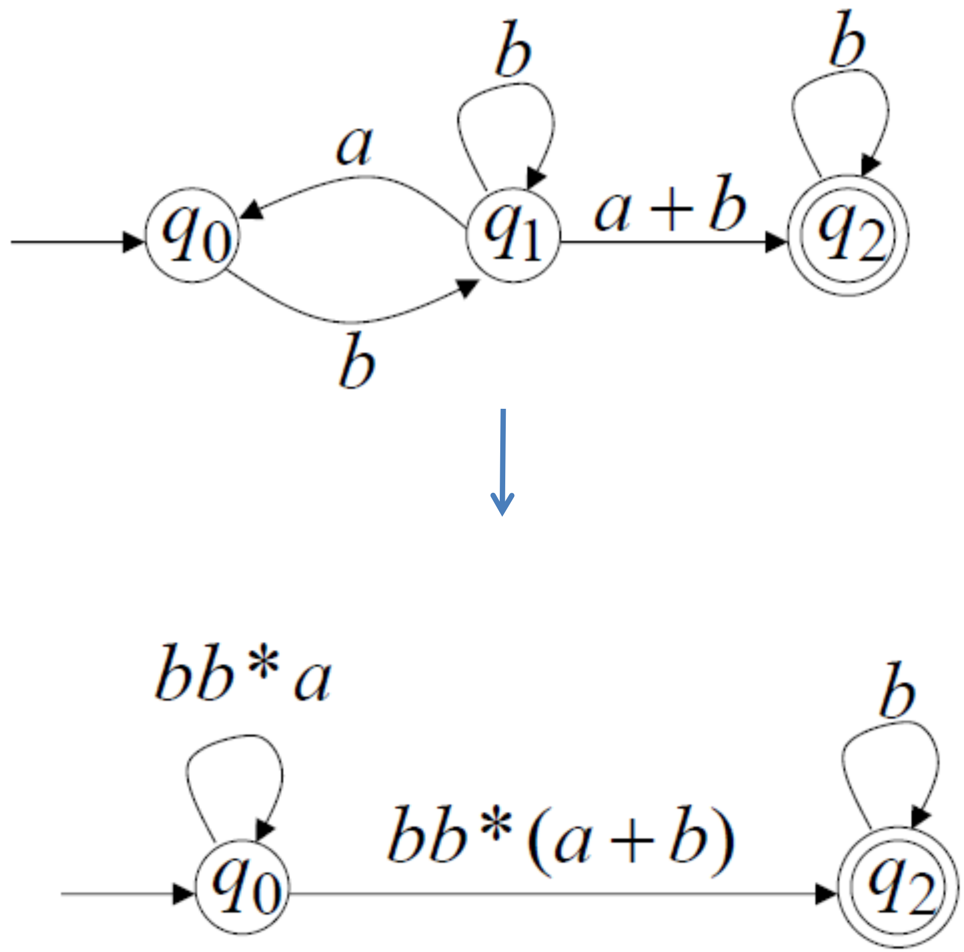



Initial Step:



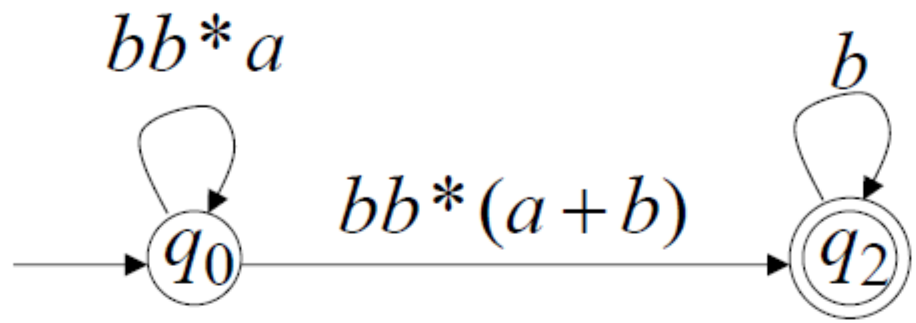
# Converting a FA into an equivalent regular expression – Example

Reducing states:



# Converting a FA into an equivalent regular expression – Example

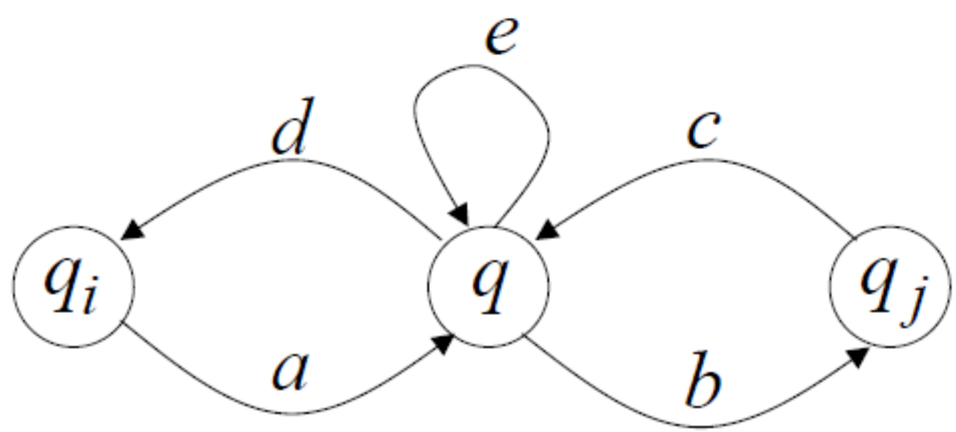
Resulting Regular Expression



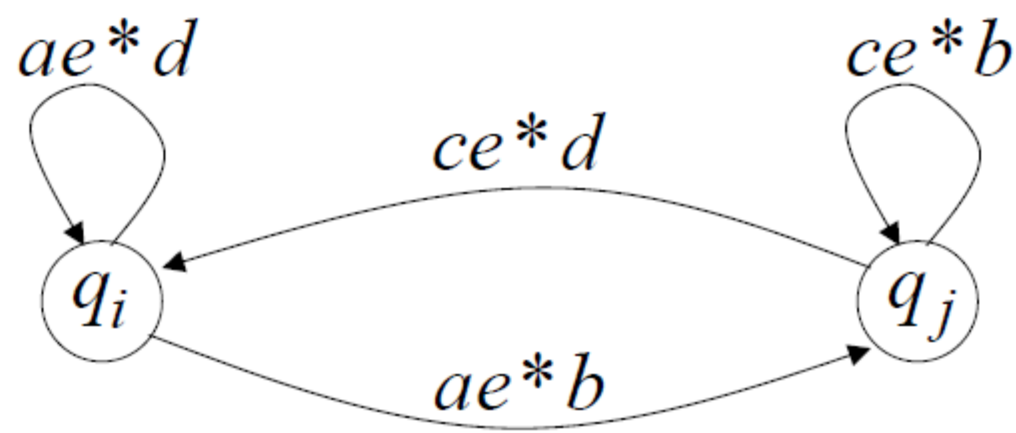

r = (bb*a)*bb*(a+b)b*

L(r)=L(M)

# Converting a FA into an equivalent regular expression – In general

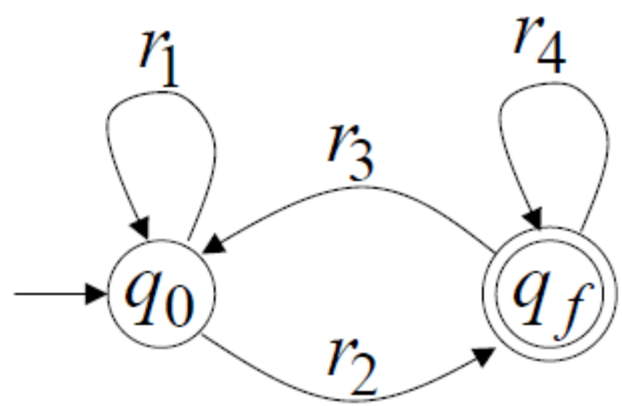



Reducing States



# Converting a FA into an equivalent regular expression – In general

The final transition graph




The resulting regular expression

$$r=r_1{}^*r_2(r_4+r_3r_1{}^*r_2)^*$$

L(r)=L(M)

# Converting a FA into an equivalent regular expression - Example