Regular Expressions

- Regular expressions (RegExps) are patterns that represent sets of words, i.e., languages
- Basic Notations:
  - For a character $c \in \Sigma$, we usually just write c instead of { c } to denote the language with the single word "c".
  - The symbol ε is used both for the "empty word", and the language containing the "empty word"
  - The empty language is represented as { }, but is rarely used in practice

Regular Expression Operators

- Three operators can be used to build up larger RegExp's from smaller RegExp's:
  - Concatenation: *A B* (*A* word, followed by a *B word*)
  - Alternation: *A* | *B* (either an *A* word or a *B word*)
  - Kleene Star: *A** (zero or more A words, concat'ed)
- We will also use parenthesis in our RegExp's to clear up any possible ambiguities

Examples (with helper definitions)

- Decimal Integer Constants:

  NZDecDigit : 1|2|…|9

  DecDigit : NZDecDigit | 0

  DecIntConst : 0 | ((-|ε) NZDecDigit DecDigit*))

- C identifiers

  Alpha : a | b | … | z | A | B | … | Z | _

  Indent : Alpha (Alpha | DecDigit)*

Quick Quiz

- What are the following languages, given an alphabet Σ?
  - Σ^* a Σ^* (here a, is a character in Σ)
  - $(\Sigma\Sigma)^*$
  - $\Sigma^* \mid \{\ \}$
  - $\Sigma^* \{\ \}$

Claim: RegExps are Equivalent to DFAs/NFAs

- In other words, any language that can be recognized using a RegExp can be recognized using a DFA/NFA, and vice-versa

Remember that these two are equiv.

- Any such language is called a regular language

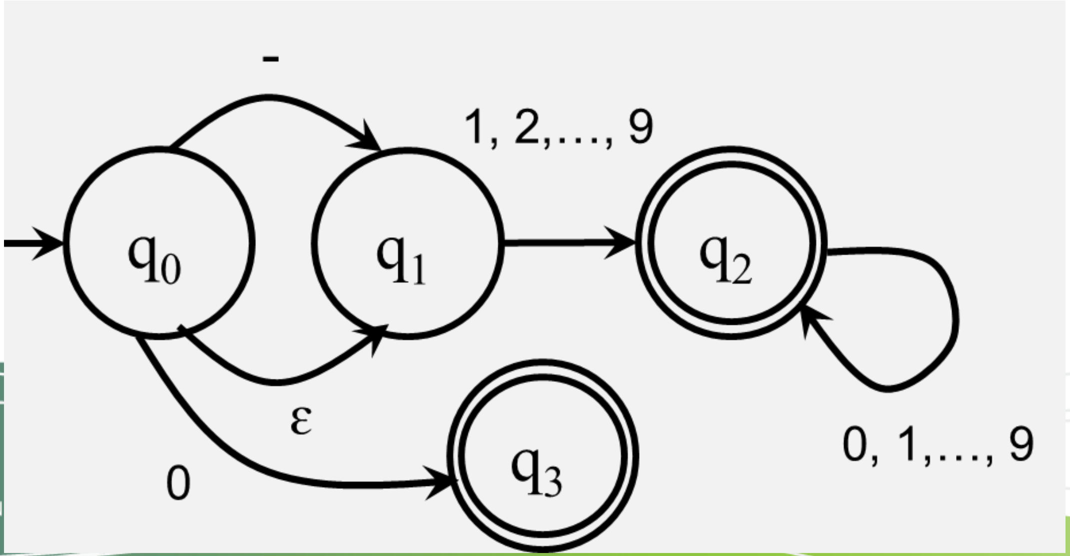
Example 1

- Decimal Integer Constants:

NZDecDigit : 1|2|…|9

DecDigit : NZDecDigit | 0

DecIntConst : 0 | ((- | ε) NZDecDigit DecDigit*))

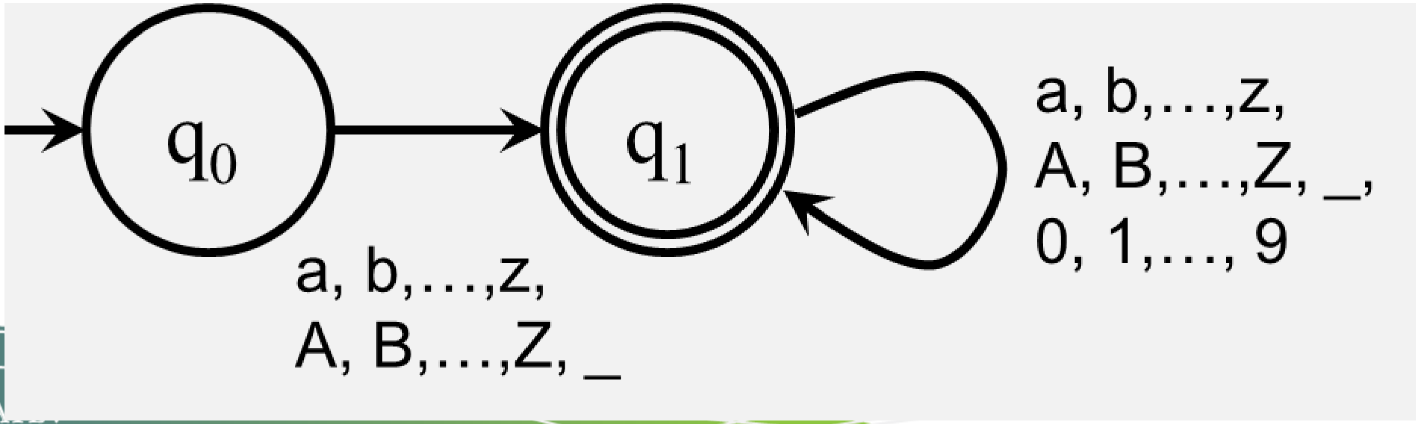


Example 2

- C identifiers

Alpha : a | b | … | z | A | B | … | Z | _

Indent : Alpha (Alpha | DecDigit)*



Proving RegExp → NFA

- Do this using structural induction on RegExp's
- Need to show for the base cases:
  - The empty word ε
  - Single characters
  - The empty language { }
- Then show for the following, given we can do it for A and B:
  - Concatenation: *A B*
  - Alternation: *A | B*
  - Kleene Star: *A**

Proving RegExp → NFA

- First three base cases should be very easy – let's do them now as a class!
- The three inductive cases are easy to see intuitively – draw NFA diagrams to "prove" that they are also true

Proving DFA → RegExp

- See proof of Lemma 1.60 from Sipser for details
- Main Idea: Use Generalized NFAs (GNFAs) which can have RegExp labels for the transitions

1. Convert the DFA to a GNFA
2. "Compact" the GNFA into a 2-state GNFA
3. Use the RegExp label on the remaining transition