NFAs and Regular Expressions.

CS154
Chris Pollett
Feb. 13, 2006.

Outline

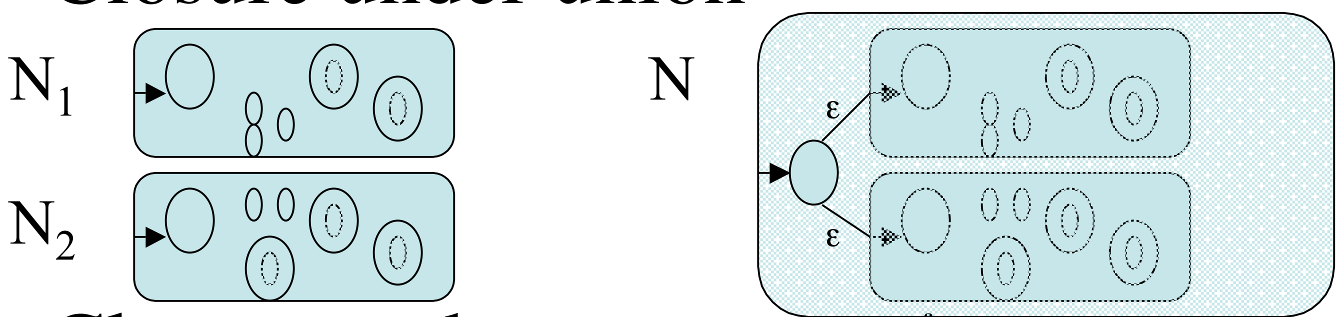
- Closure Properties of NFAs
- Regular Expression
- Equivalence with Finite Automata

Corollary of NFA-DFA equivalence

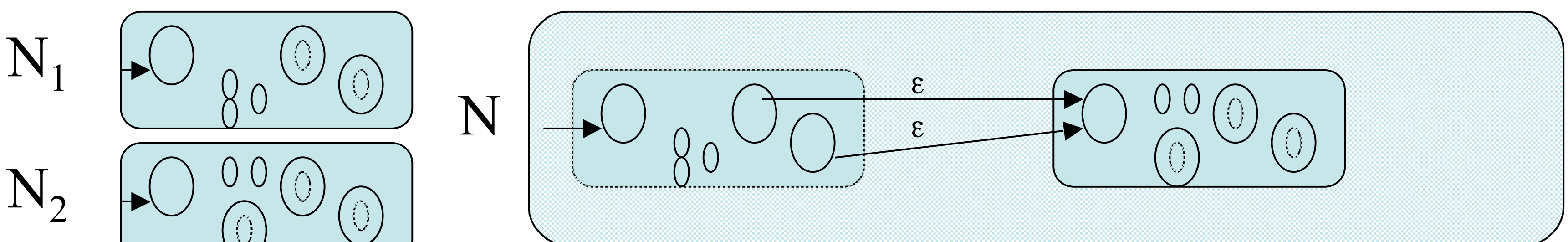
- Every DFA is trivially an NFA.
- Last day, we showed given an NFA how to construct a DFA recognizing the same language.
- Therefore, we get that a language is regular if and only if it is recognized by some NFA.

NFA based proofs of Closure Properties of Regular Languages

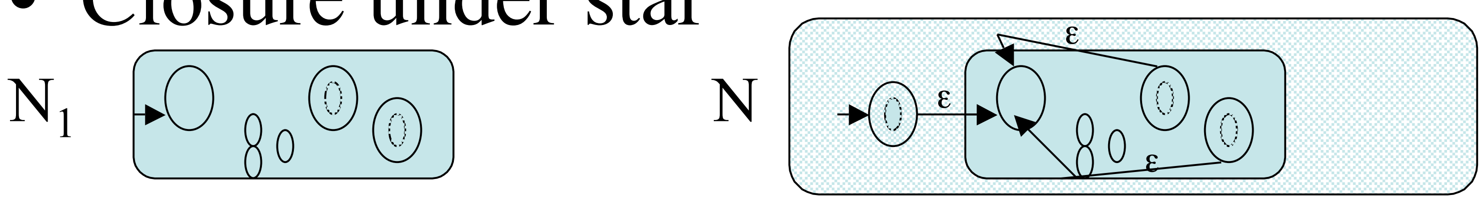
- Closure under union



- Closure under concatenation



- Closure under star



Introduction to Regular Expressions

- In arithmetic, we can use the operations + and * to build up expressions such as:
  (5 + 3) * 4.
- Similarly we can use the regular operations to build up expressions describing regular languages.
- For instance, $0(0\cup 1)^*$ (We use juxtaposition to abbreviate concatenation: $0\circ(0\cup 1)^*$).
- This means the language which results from concatenating the language containing 0 with the language of $(0\cup 1)^*$. This in turn is the star of the union of the two languages one containing just 0; the other containing just 1.
- These kind of expressions are used in many modern programming languages: Perl, PHP, Java, AWK, GREP,

Formal Definition of a Regular Expression

- We say that R is a regular expression if R is
  1. a for some symbol a in the alphabet Σ,
  2. ε
  3. $\emptyset$
  4. $(R_1 \cup R_2)$ where R_1 and R_2 are regular expressions
  5. $(R_1 \circ R_2)$ where R_1 and R_2 are regular expressions
  6. $(R_1)^*$ where R_1 is a regular expression
- We write R^+ as a shorthand for RR^*.
- We write L(R) for the language given by the regular expression

Examples of the Definition

- $0^*1\,0^* = \{w \mid w \text{ contains a single } 1\}$
- $(01 \cup 10) = \{01, 10\}$
- $(\Sigma\Sigma)^* = \{w \mid w \text{ is of even length}\}$
- $(\varepsilon \cup 0)(\varepsilon \cup 1) = \{\varepsilon, 0, 1, 01\}$
- $1^* \varnothing = \varnothing$
- $\varnothing^* = \{\varepsilon\}$

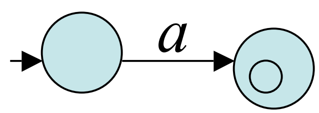
Equivalence with Finite Automata

- We want to show that a language is regular if and only if some regular expression describes it.
- We will do this in two steps:
  - Prove if a language is described by a regular expression, then it is regular
  - Prove if a language is regular, then it is described by a regular expression.

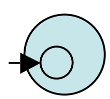
Proof that regular expression implies regular

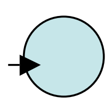
- It suffices to come up with NFAs for the three languages (1), (2), (3) a couple slides back since we already know the regular languages are closed under union, concatenation and *.

1. Let R= *a* for some *a* in Σ. Then the following NFA recognizes the languages contain only *a*.



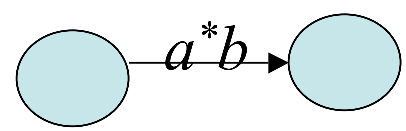
2. Let R= ε. Then the following NFA recognizes it:



3. Let R= $\emptyset$. Then the following NFA recognizes it: 

Proof that regular implies the language of some regular expression

- We will again split the proof into two parts:
  - We first define a new kind of finite automata called a generalized nondeterministic finite automata (GNFA) and show how to convert any DFA into a GNFA.
  - Then we show how to convert any GNFA into a regular expression.
- To begin we define a GNFA to be an NFA where we allow transition arrows to have any regular expression as labels:



Converting DFAs to GFNA

- We will be interested in GNFAs that have the following special form:
  - The start state has transition arrows to every other state but no arrows coming in from other states.
  - There is a single accept state, and it has arrows coming in from every other state but no arrows going to any other state.
  - Except for the start and accepts state, one arrow goes from from every state to every other state and also from each state to itself.
- To convert a DFA into a GNFA, we add a new start state with and ε arrow to the old start state and a new accept state with ε arrows from the old accept states.
- If any arrows have multiple labels (or if we have two or more arrows between the same two states) we replace each with a single label whose label is the union labels of the these arrows.
- Finally, we add arrows with labels $\emptyset$ between states which had no labels so as to satisfy the remaining conditions of our special form.