Expressões Regulares

Definições
Equivalência para Autômatos Finitos

Tradução dos slides do Prof. Jeffrey D. Ullman (Stanford University)

RE's: Introdução

- *Expressões Regulares* são um maneira algébrica de descrever linguagens.
- As RE's descrevem exatamente as mesmas linguagens que as diversas formas de autômatos: as linguagens regulares.
- Se E é uma expressão regular, então L(E) é a linguagem que ela define.

Operadores de RE's

- *União* (L∪M): conjunto de strings que estão em L ou M, ou em ambas.
- *Concatenação* (L.M): conjunto de strings que podem ser formados tomando-se qualquer string em L e concatenando-se esse string com qualquer string em M.
- *Fechamento* (L*): conjunto de strings que podem ser formados tomando-se qualquer número de strings de L e concatenando-se todos eles.

RE's: Definição

- Podemos descrever as expressões regulares recursivamente.

- Não só descrevemos quais as RE's válidas mas, para cada RE E, descrevemos a linguagem que ela representa (L(E)).

RE's: Definição – (2)

- Base 1: Se *a* é qualquer símbolo, então **a** é uma RE, e L(**a**) = {a}.
  - Note: {a} é a linguagem contendo uma string, e esta string é de comprimento 1.
- Base 2: ϵ é uma RE, e L(ϵ) = {ϵ}.
- Base 3: ∅ é uma RE, e L(∅) = ∅.

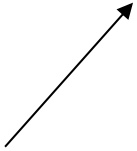
RE's: Definição – (3)

- Indução 1: Se E_1 e E_2 são expressões regulares, então E_1+E_2 é uma expressão regular, e $L(E_1+E_2) = L(E_1) \cup L(E_2)$.
- Indução 2: Se E_1 e E_2 são expressões regulares, então E_1E_2 é uma expressão regular, e $L(E_1E_2) = L(E_1)L(E_2)$.

Concatenação : o conjunto de strings wx tal que w está em $L(E_1)$ e x está em $L(E_2)$.

RE's: Definição – (3)

- Indução 3: Se E é uma RE, então E* é uma RE, e L(E*) = (L(E))*.



Fechamento, or "fechamento de Kleene" = conjunto de strings $w_1w_2...w_n$, para algum $n \geq 0$, onde cada w_i está em L(E).
Note: quando n=0, a string é ϵ.

Precedência de Operadores

- A ordem de precedência é:
  - * (mais alta)
  - . concatenação
  - + (mais baixa).

- Parênteses podem ser usados com a finalidade de agrupar operandos exatamente como pretendemos.

Exemplos: RE's

- L(**01**) = {01}.
- L(**01**+**0**) = {01, 0}.
- L(**0**(**1**+**0**)) = {01, 00}.
  - Note a ordem de precedência dos operadores.
- L(**0***) = {ϵ, 0, 00, 000,… }.
- L((**0**+**10**)*(ϵ+**1**)) = todos strings de 0's e 1's sem dois 1's consecutivos.

Exercícios: RE's

Crie expressões regulares $\Sigma = \{0,1\}$:

1. {w | w contém um único 1}
2. {w | w contém pelo menos um 1}
3. {w | w contém a string 001 como substring}
4. {w | |w| é par}
5. {w | |w| é um múltiplo de 3}
6. {w | 6º símbolo direita/esquerda é 1}

Equivalência de RE's e Autômatos

Devemos mostrar que:

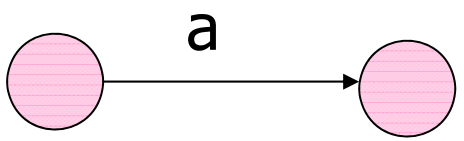
- Toda linguagem definida por uma expressão regular também é definida por um autômato.
- Toda linguagem definida por um autômato também é definida por uma expressão regular.

Conversão de RE em ϵ-NFA

- Prova é uma indução sobre o número de operadores (+, concatenação, *) no RE.

RE para ϵ-NFA: Base

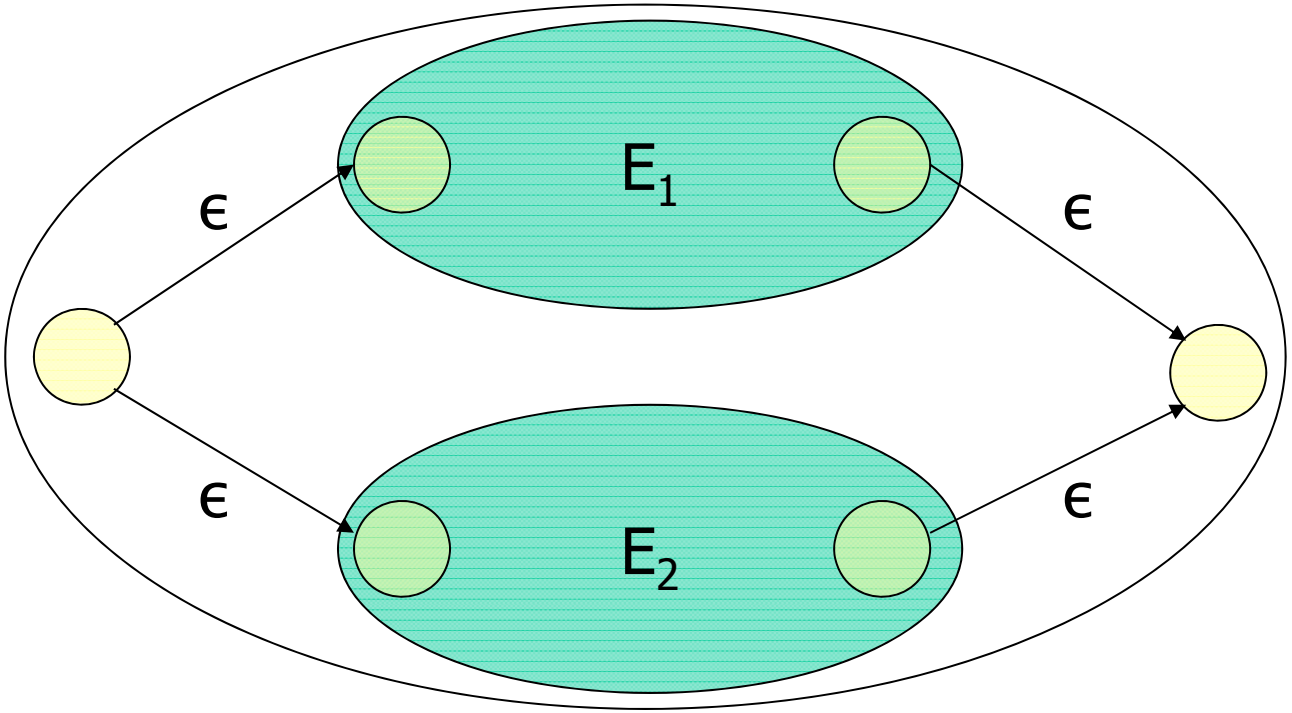
- Símbolo **a**:



- ϵ:

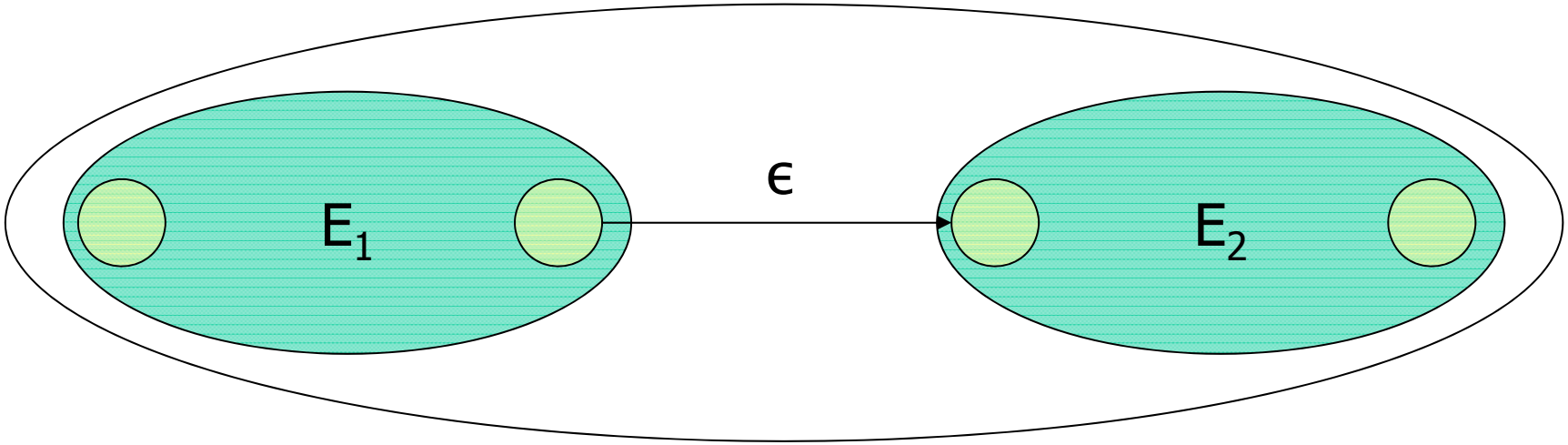
- ∅:

RE para ϵ-NFA: Indução 1 – União



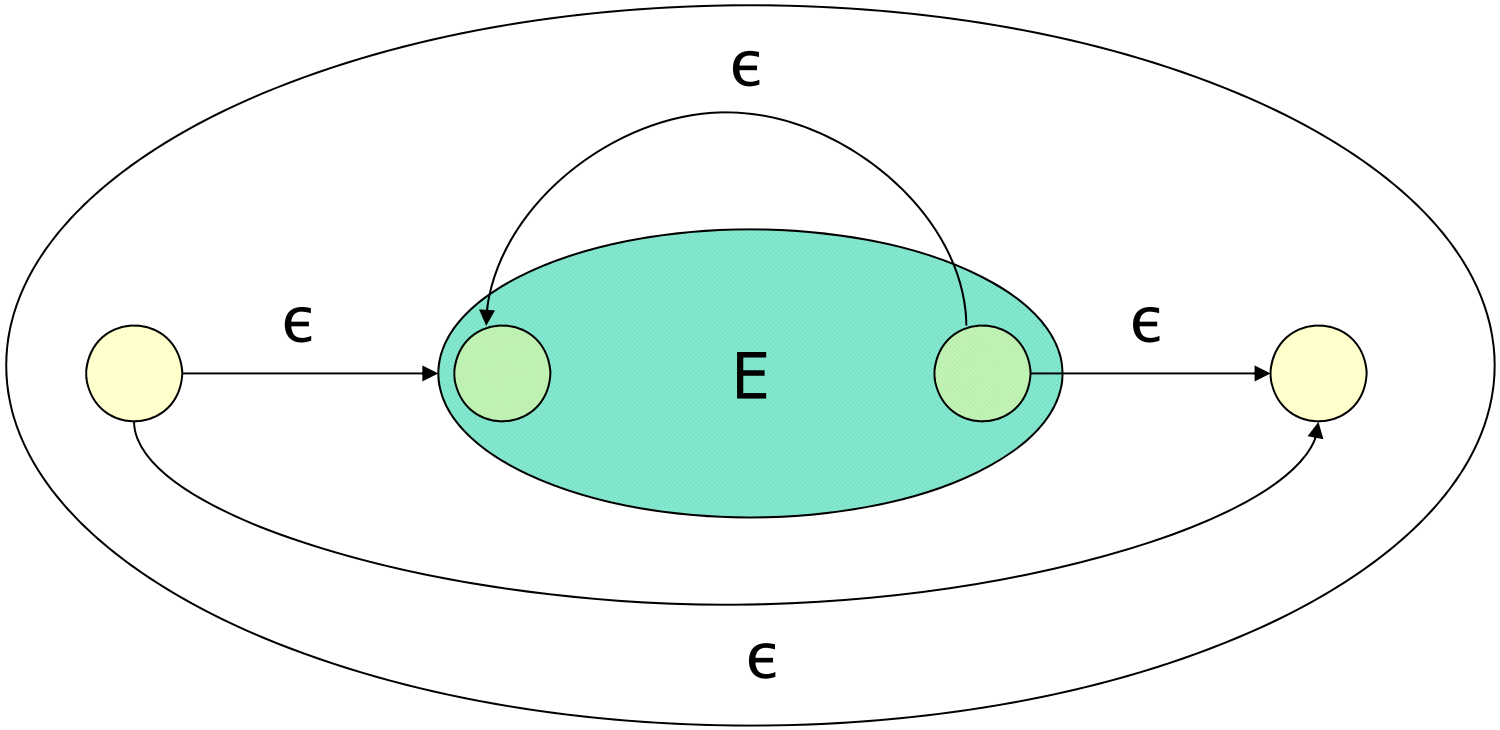
$E_1 \cup E_2$

RE para ϵ-NFA: Indução 2 – Concatenação



E_1E_2

RE para ϵ-NFA: Indução 3 – Fechamento



E*

DFA-para-RE

- Calcular os caminhos de um DFA.

  ou

- Eliminação de estados.

(Seções 3.2.1 e 3.2.2 do livro do Hopcroft).

Resumo

- Cada tipo de autômato estudado (DFA, NFA, ϵ-NFA), e as expressões regulares, definem exatamente o mesmo conjunto de linguagens: linguagens regulares.

Leis Algébricas para RE's

- União e concatenação são semelhantes a adição e multiplicação, respectivamente.
  - + é comutativa and associativa;
  - concatenação é associativa.
  - Concatenação distribui sobre +.
  - Exceção: Concatenação **não** é comutativa.

Identidades e Aniquiladores

- $\emptyset$ é a identidade para +.
  - $R + \emptyset = R$.
- ϵ é a identidade para concatenação.
  - $\epsilon R = R\epsilon = R$.
- $\emptyset$ é o aniquilador para a concatenação.
  - $\emptyset R = R\emptyset = \emptyset$.

Exercícios: RE's

Escreva expressões regulares correspondentes às seguintes linguagens:

1. Conjunto de strings sobre o alfabeto {a,b,c} que contém pelo menos um a e pelo menos um b.
2. Conjunto de todos os strings de 0's e 1's tais que todo par de 0's adjacentes aparece antes de qualquer par de 1's adjacentes.
3. O conjunto de todos strings que não contêm o substring 101.
4. O conjunto de todos os strings que têm no máximo um par de 0's consecutivos ou um par de 1's consecutivos.