Tema 5

Inferência gramatical para linguagens regulares

Professora:
Ariane Machado Lima

Inferência gramatical

- Aprendizado/inferência de uma gramática a partir de … algum tipo de informação. Ex:
  - *Texto* (exemplos positivos)
  - *Informante* (que pode te dar exemplos positivos e negativos)

Inferência gramatical - Paradigmas

- Identificação no limite
  - Objetivo: identificar uma linguagem alvo $\mathbb{L}$
  - G_i = gramática inferida a partir de uma amostra $S^+ = \{x_1, x_2, \ldots, x_i\}$
  - Para j >= t exemplos, G_j não muda e $L(G_j) = \mathbb{L}$

Inferência gramatical - Paradigmas

- Identificação no limite
- Alguns resultados:
  - **Nenhuma** classe de linguagens super-finita (que contém todas as finitas e pelo menos uma infinita) pode ser identificada no limite apenas a partir de ***texto***
  - Qualquer classe de linguagens recursivas **é** identificável no limite a partir de um ***informante***

Inferência gramatical - Paradigmas

- Aprendizado PAC (Provavelmente Aproximadamente Correto)
  - Objetivo: ter grande chance de inferir (a partir de uma amostra) uma G que seja uma boa aproximação da linguagem alvo $\mathbf{L}$

Inferência gramatical para linguagens regulares

- Aprendizado de autômatos
- Autômato inicial que reconhece a amostra → generalização via junções de estados (*merges*)

Inferência gramatical para linguagens regulares

- RPNI: AFD (identificação no limite a partir de um informante)
- LAPFA: autômato probabilístico (aprendizado PAC a partir de uma amostra positiva)

RPNI

RPNI

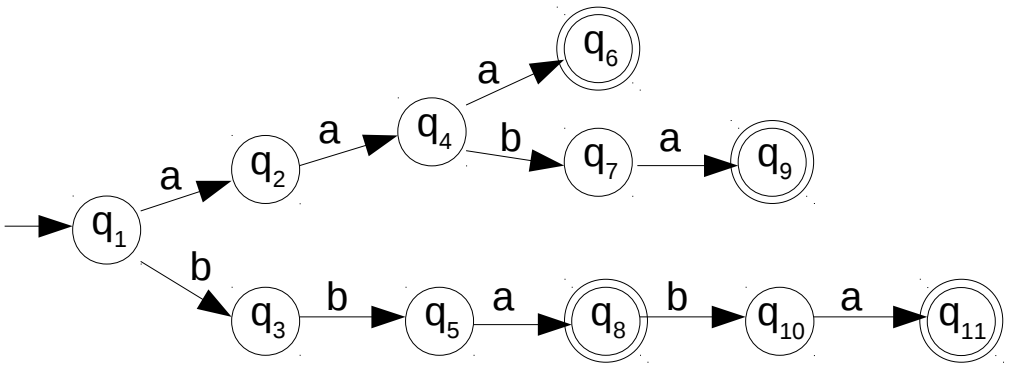
- RPNI: Regular Positive and Negative Inference (ONCINA & GARCIA, 1992)
- Aprende no limite…
- … um AFD (note que não probabilístico)
- … baseado em uma amostra positiva (S^+) e uma negativa (S^-)…
- … em tempo polinomial

RPNI

- DFA **possivelmente** com uma definição ligeiramente diferente (HIGUERA, 2010):
- δ(q,s) pode não estar definido
- Há dois tipos de estados finais:
  - F_A: estados finais de aceitação
  - F_R: estados finais de rejeição
- Se a cadeia x termina em um estado q = $\delta(q_0, x)$
  - Se q ϵ F_A a cadeia é aceita
  - Se q ϵ F_R a cadeia é rejeitada
  - Se q $\not\epsilon$ (F_A U F_R) nada se diz sobre a cadeia
- Obs: $\delta(q_0, x)$, x sendo uma cadeia, indica o estado em que se chega após sucessivas transições partindo de q_0 e processando x símbolo a símbolo

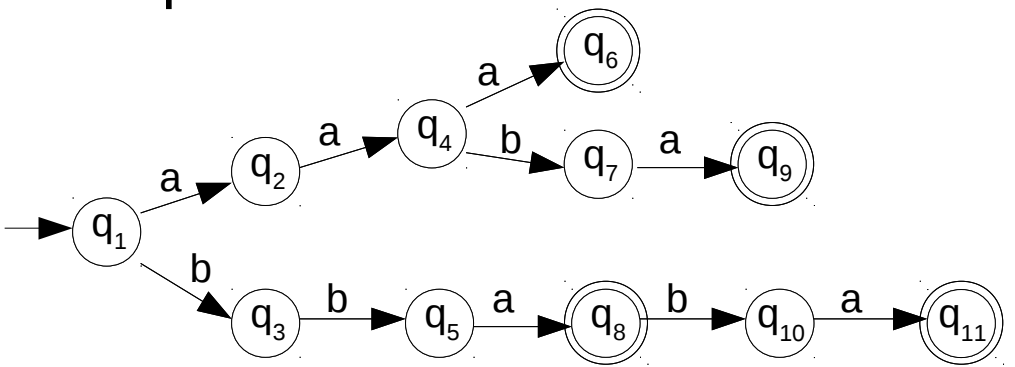
RPNI (ideia geral)

- Ponto de partida: PTA (*prefix tree acceptor*) da amostra positiva
- Ex: S^+ = {aaa, aaba, bba, bbaba}



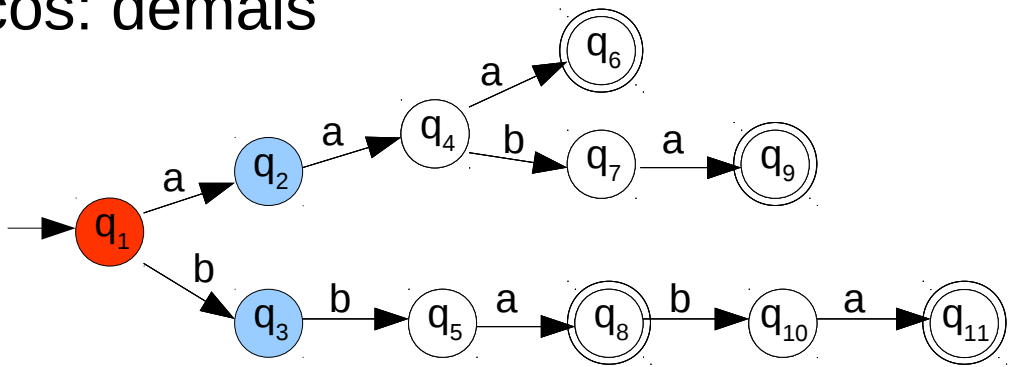
RPNI (ideia geral)

- Ponto de partida: PTA (*prefix tree acceptor*) da amostra positiva
- Processa estados em ordem de busca em largura
  - Tenta juntar q com q’
  - Se nenhuma cadeia de S- for aceita, mantém junção
  - Senão, tente outro q’



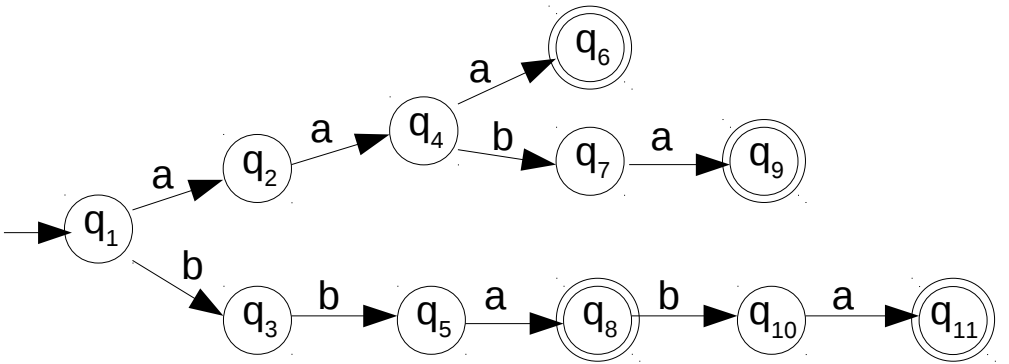
Coloração de estados

- Para executar a busca em largura:
  - Estados vermelhos: já analisados (não serão revisitados)
  - Estados azuis: candidatos à próxima análise para serem ou não unidos com um estado vermelho (sucessores de vermelhos)
  - Estados brancos: demais



RPNI (ideia geral)

- Ponto de partida: PTA (*prefix tree acceptor*) da amostra positiva
- Processa estados em ordem de busca em largura
  - Tenta juntar um q com um q'
  - Se nenhuma cadeia de S- for aceita, mantém junção
  - Senão
    - tente outro q'
    - Senão achar, q vira q (promoção)



RPNI - Algoritmo

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

A ← ***merge***(A, q_r, q_b)

estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

senão ***promove***(A, q_b)

Para cada q

Se q é o estado final de uma cadeia de S^- então F_R ← F_R U {q}

RPNI - Algoritmo

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

A ← ***merge***(A, q_r, q_b)

estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

senão ***promove***(A, q_b)

Para cada q

Se q é o estado final de uma cadeia de S^- então F_R ← F_R U {q}

Por ex: escolher na ordem crescente do rótulo do estado

RPNI - Algoritmo

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

A ← ***merge***(A, q_r, q_b)

estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

senão ***promove***(A, q_b)

Para cada q

Se q é o estado final de uma cadeia de S^- então F_R ← F_R U {q}

Uma cadeia da S^- não pode ser aceita

RPNI - Algoritmo

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

A ← ***merge***(A, q_r, q_b)

estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

senão ***promove***(A, q_b)

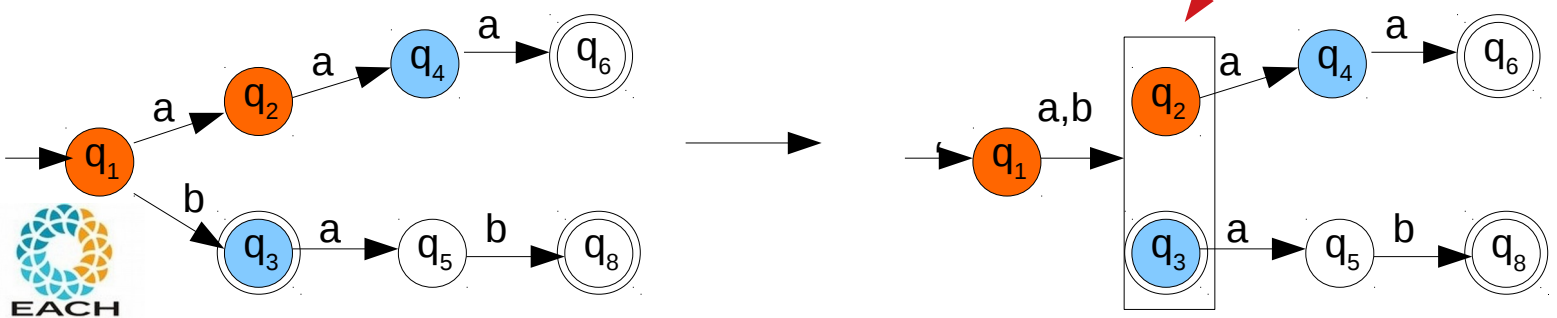
Para cada q

Se q é o estado final de uma cadeia de S^- então F_R ← F_R U {q}

merge (A, q, q')

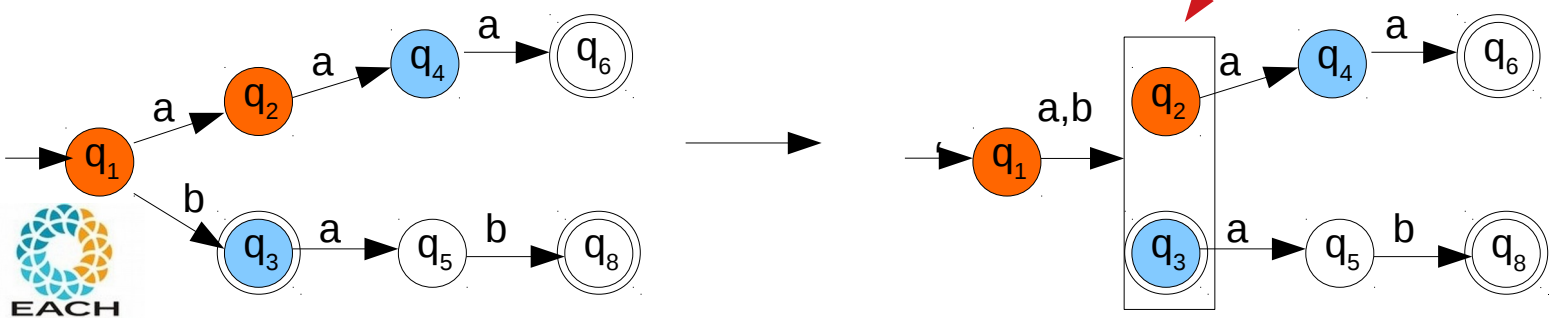
Obs: esse NÃO é o mesmo AFD do exemplo anterior

Ex: merge (A, q_2, q_3)



merge (A, q, q')

Ex: merge (A, q_2, q_3)



OPS!
Não-determinismo!

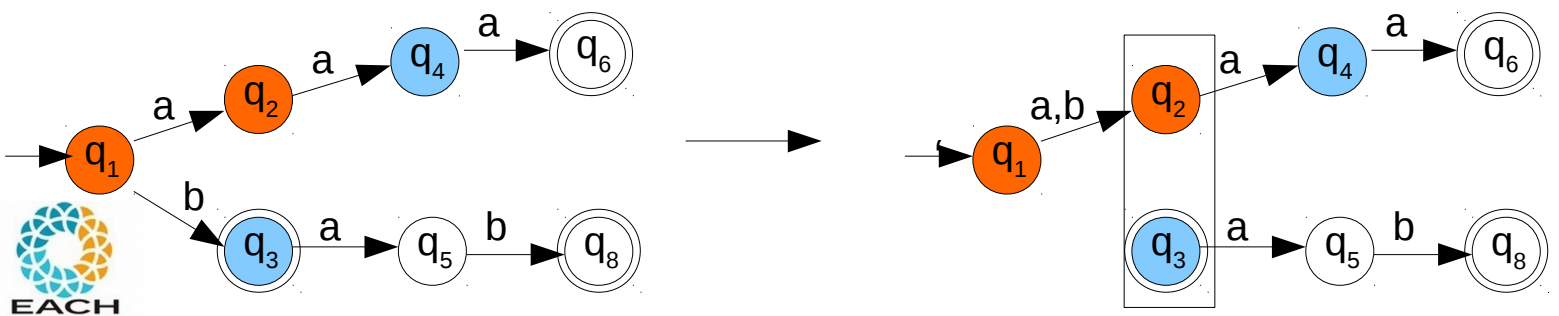
merge (A, q, q')

para todo par (q_i, s) tal que $\delta(q_i,s) = q'$

$\delta(q_i,s) \leftarrow q$

retorna ***fold***(A, q, q')

Ex: merge (A, q_2, q_3)



fold (A, q, q')

Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$

para todo s ϵ Σ

  Se δ(q',s) está definido

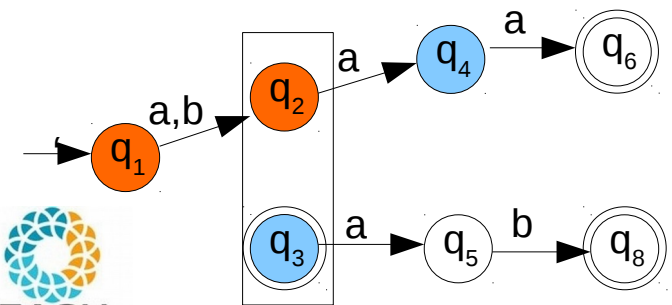
    Se δ(q,s) está definido

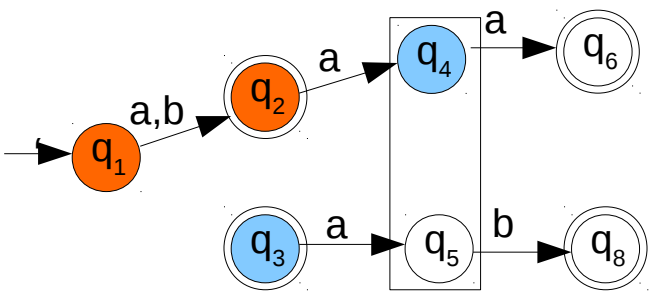
      A ← fold (A, δ(q,s), δ(q',s))

    Senão δ(q,s) ← δ(q',s)

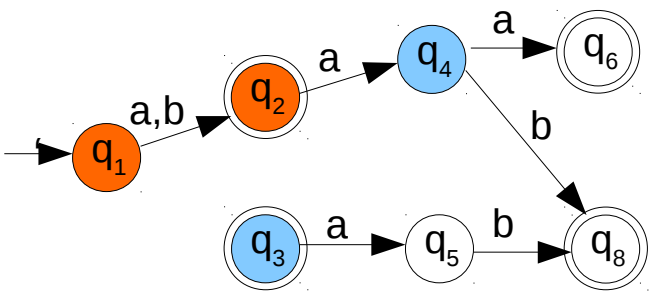
retorna A

Ex: fold (A, q_2, q_3)





fold (A, q_4, q_5)



fold (A, q, q')

Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$

para todo s ϵ Σ

Se δ(q',s) está definido

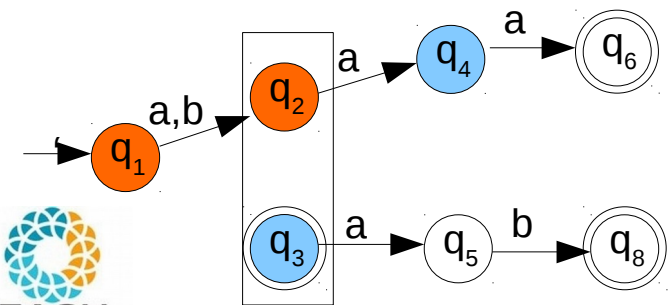
Se δ(q,s) está definido

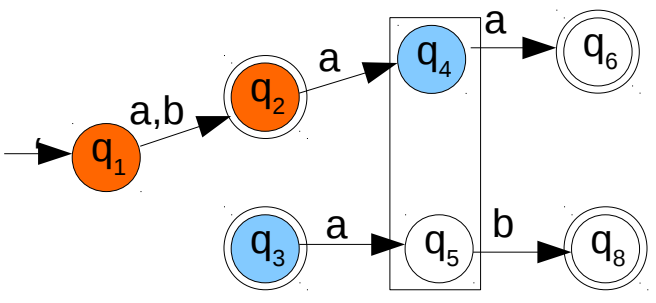
A ← fold (A, δ(q,s), δ(q',s))

Senão δ(q,s) ← δ(q',s)

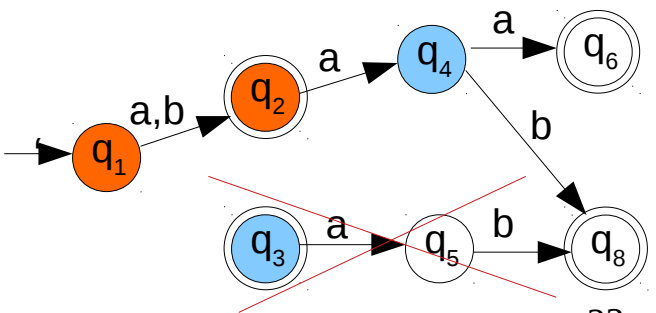
retorna A

Ex: fold (A, q_2, q_3)





fold (A, q_4, q_5)



RPNI - Algoritmo

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

A ← ***merge***(A, q_r, q_b)

estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

senão ***promove***(A, q_b)

q_b se torna vermelho, e seus próximos estados tornam-se azuis

Para cada q

Se q é o estado final de uma cadeia de S^- então F_R ← F_R U {q}

RPNI - Exemplo

Exemplo - RPNI

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

A ← ***merge***(A, q_r, q_b)

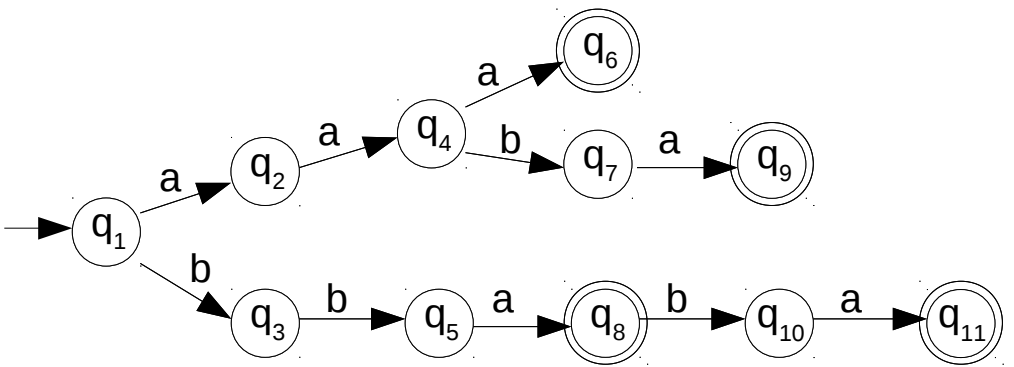
estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

senão ***promove***(A, q_b)

Para cada q

Se q é o estado final de uma cadeia de S^- então $F_R \leftarrow F_R \cup \{q\}$

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$



Exemplo - RPNI

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

A ← ***merge***(A, q_r, q_b)

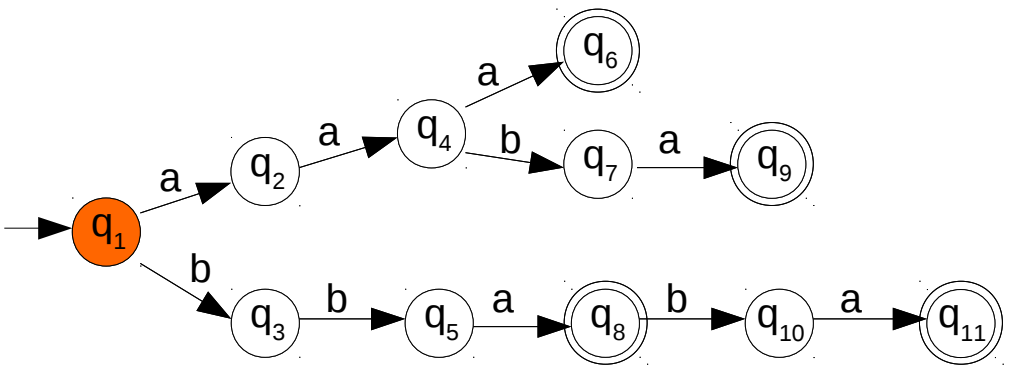
estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

senão ***promove***(A, q_b)

Para cada q

Se q é o estado final de uma cadeia de S^- então $F_R \leftarrow F_R \cup \{q\}$

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$



Exemplo - RPNI

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

A ← ***merge***(A, q_r, q_b)

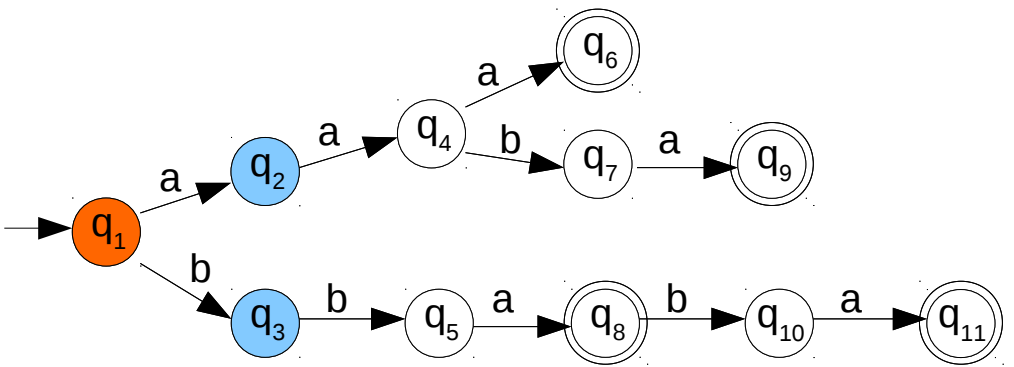
estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

senão ***promove***(A, q_b)

Para cada q

Se q é o estado final de uma cadeia de S^- então F_R ← F_R U {q}

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$



Exemplo - RPNI

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

A ← ***merge***(A, q_r, q_b)

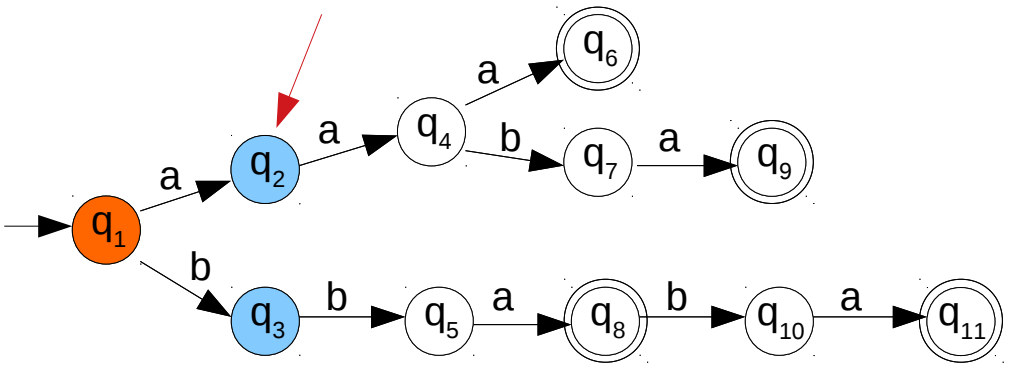
estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

senão ***promove***(A, q_b)

Para cada q

Se q é o estado final de uma cadeia de S^- então F_R ← F_R U {q}

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$



Exemplo - RPNI

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

A ← ***merge***(A, q_r, q_b)

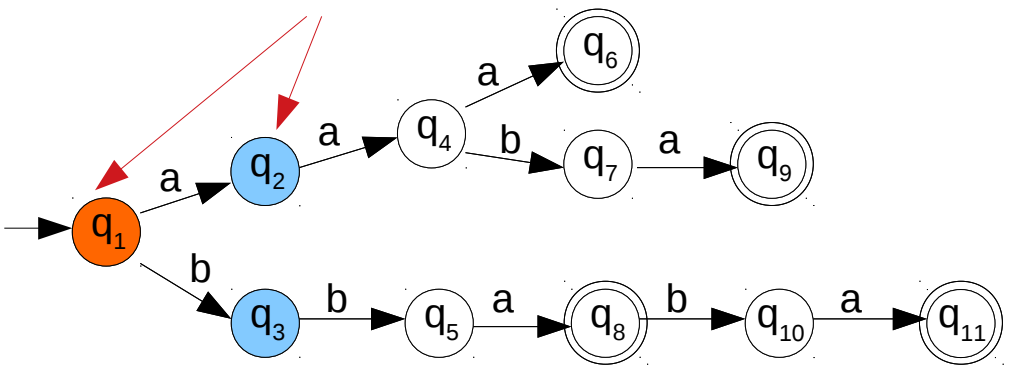
estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

senão ***promove***(A, q_b)

Para cada q

Se q é o estado final de uma cadeia de S^- então F_R ← F_R U {q}

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$

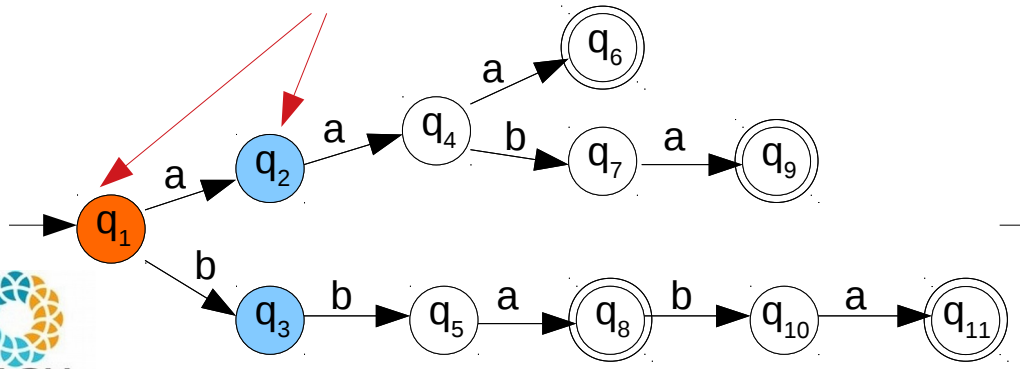


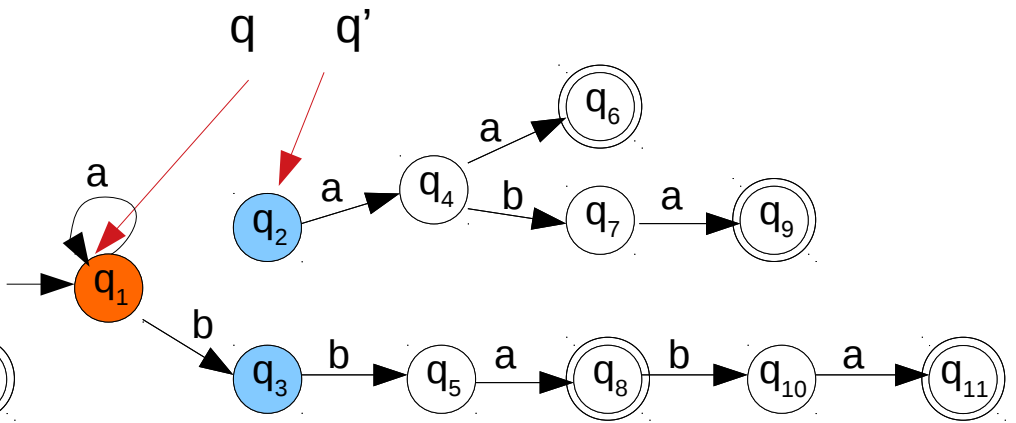
Exemplo - RPNI

merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se δ(q',s) está definido
        Se δ(q,s) está definido
            A ← fold (A, δ(q,s), δ(q',s))
        Senão δ(q,s) ← δ(q',s)
retorna A

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$



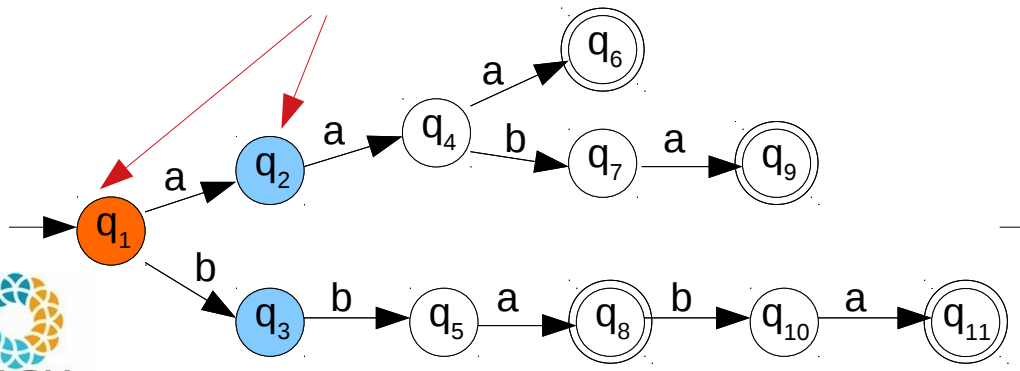


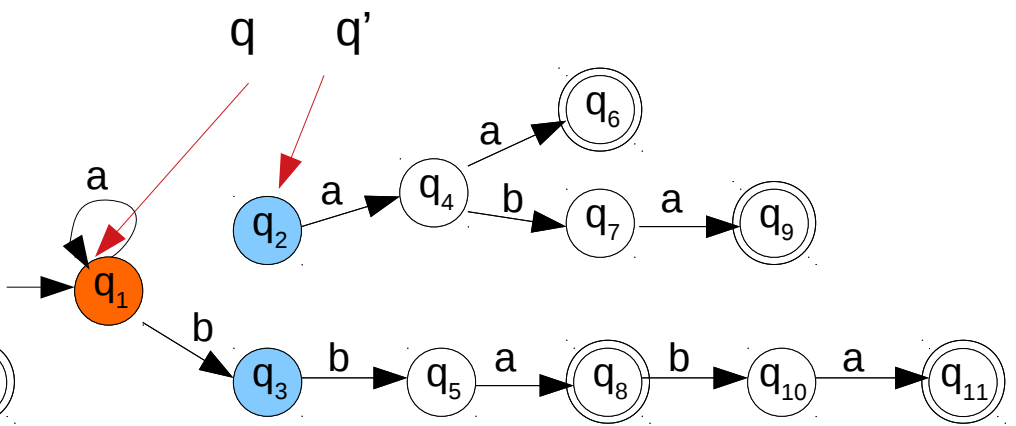
Exemplo - RPNI

merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$

fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se $\delta(q',s)$ está definido
        Se $\delta(q,s)$ está definido
            A ← fold (A, $\delta(q,s)$, $\delta(q',s)$)
        Senão $\delta(q,s) \leftarrow \delta(q',s)$
retorna A



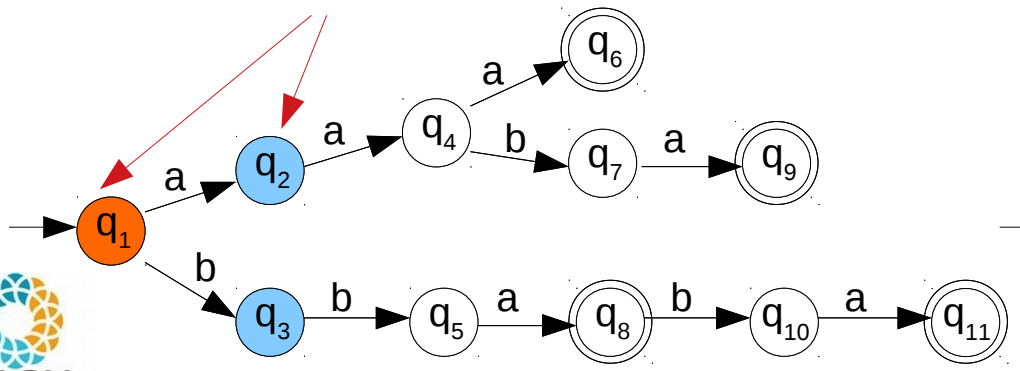


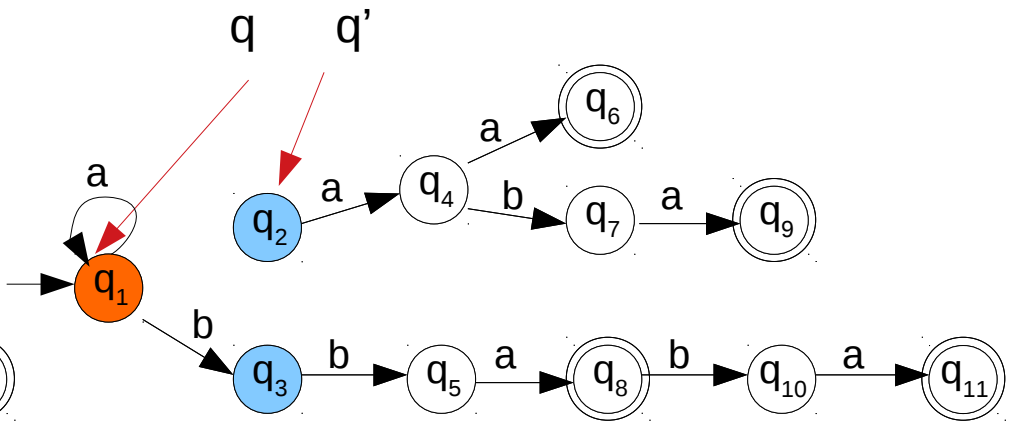
Exemplo - RPNI

merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se δ(q',s) está definido
        Se δ(q,s) está definido
            A ← fold (A, δ(q,s), δ(q',s))
        Senão δ(q,s) ← δ(q',s)
retorna A

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$





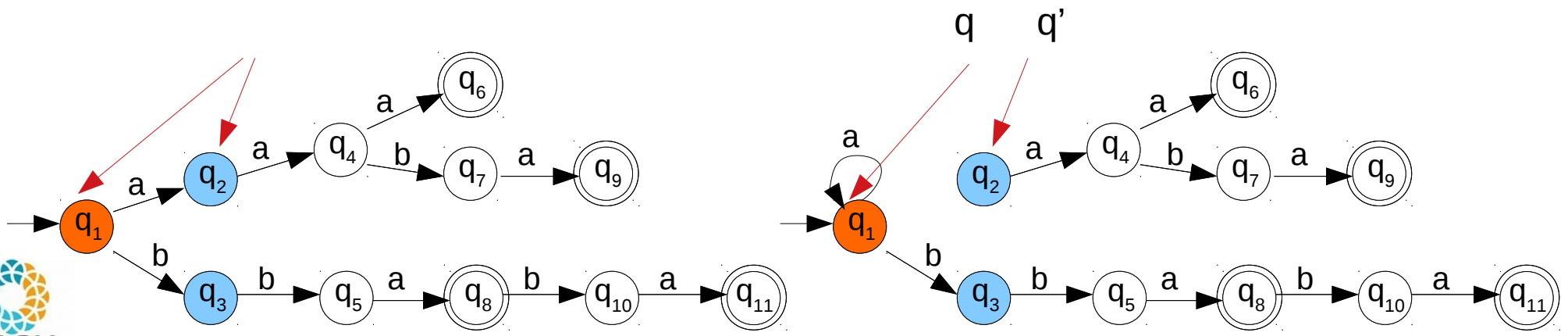
Exemplo - RPNI

merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se δ(q',s) está definido
        Se δ(q,s) está definido
            A ← fold (A, δ(q,s), δ(q',s))
        Senão δ(q,s) ← δ(q',s)
retorna A

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$

fold(A, q_1, q_4)



Exemplo - RPNI

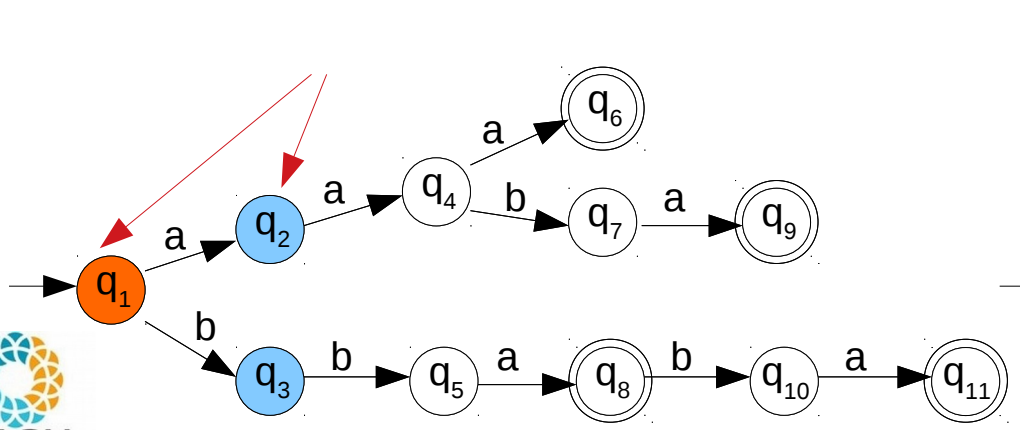
merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

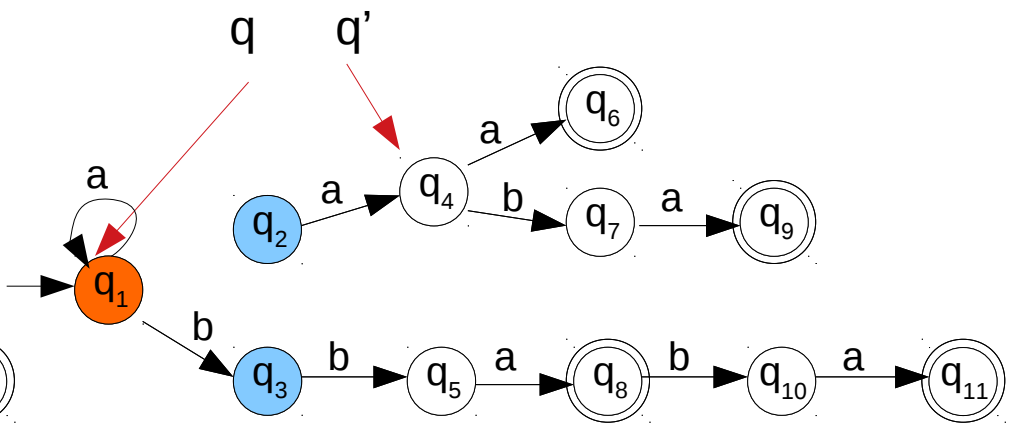
fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se δ(q',s) está definido
        Se δ(q,s) está definido
            A ← fold (A, δ(q,s), δ(q',s))
        Senão δ(q,s) ← δ(q',s)
retorna A

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$

Começando com s = a

fold(A, q_1, q_4)





Exemplo - RPNI

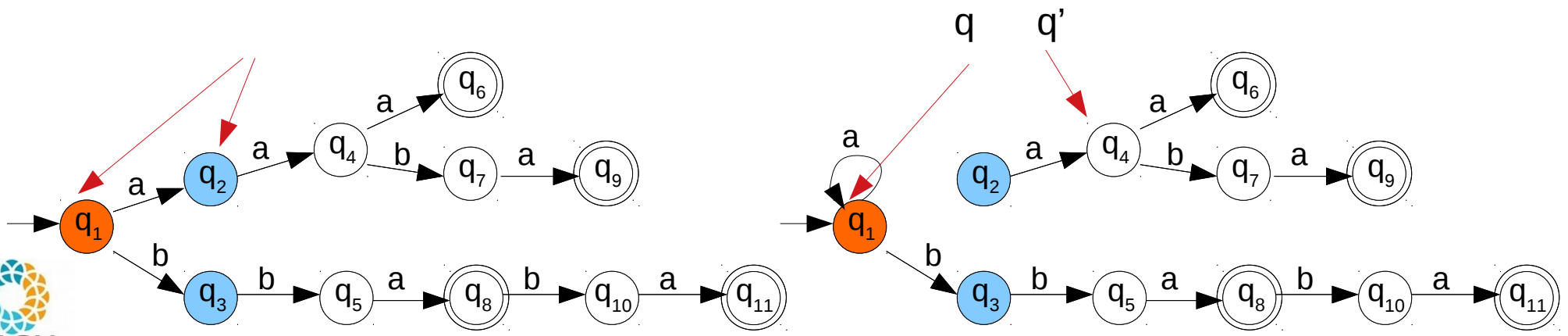
merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se δ(q',s) está definido
        Se δ(q,s) está definido
            A ← fold (A, δ(q,s), δ(q',s))
        Senão δ(q,s) ← δ(q',s)
retorna A

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$

Começando com s = a

fold(A, q_1, q_4)
    fold(A, q_1, q_6)

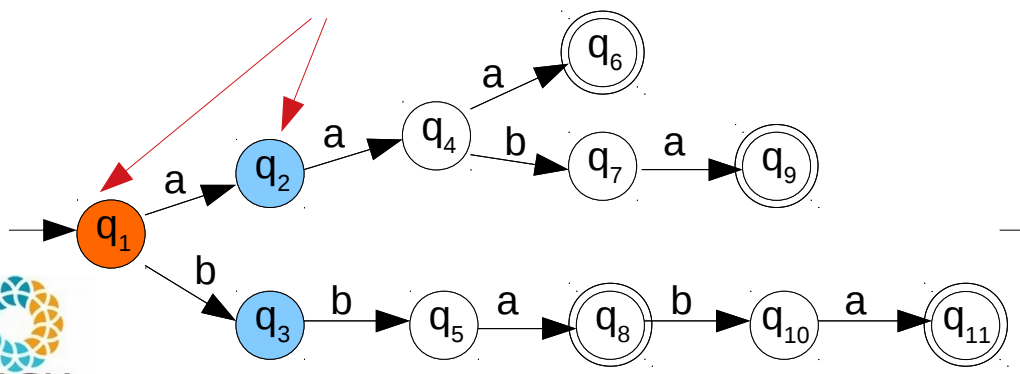


Exemplo - RPNI

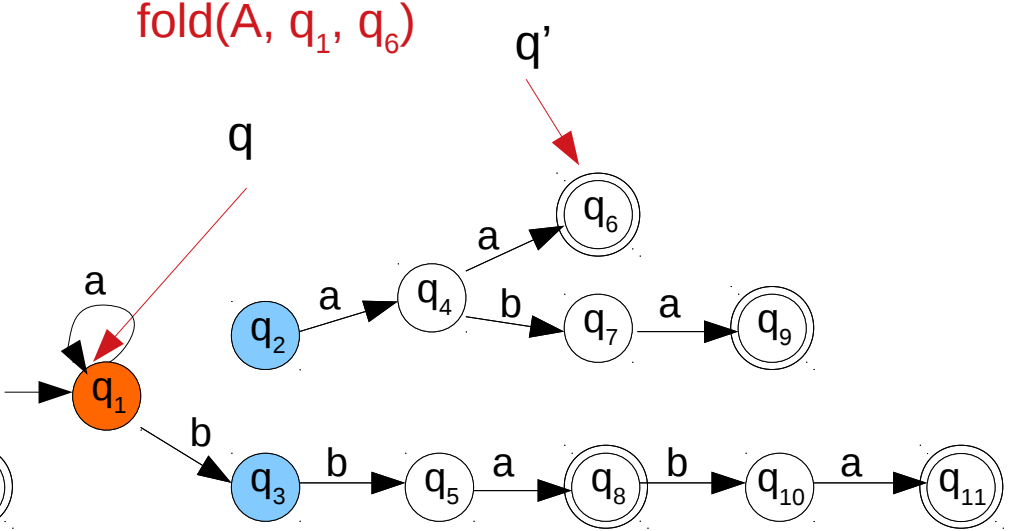
merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se δ(q',s) está definido
        Se δ(q,s) está definido
            A ← fold (A, δ(q,s), δ(q',s))
        Senão δ(q,s) ← δ(q',s)
retorna A

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$



fold(A, q_1, q_4)
fold(A, q_1, q_6)

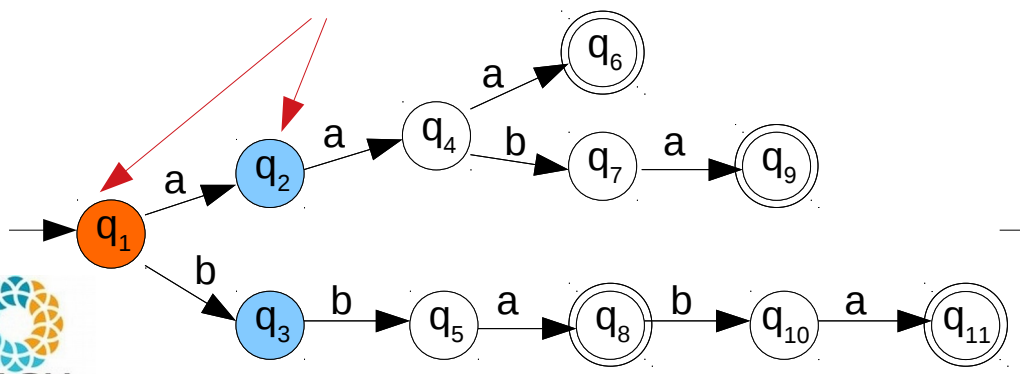


Exemplo - RPNI

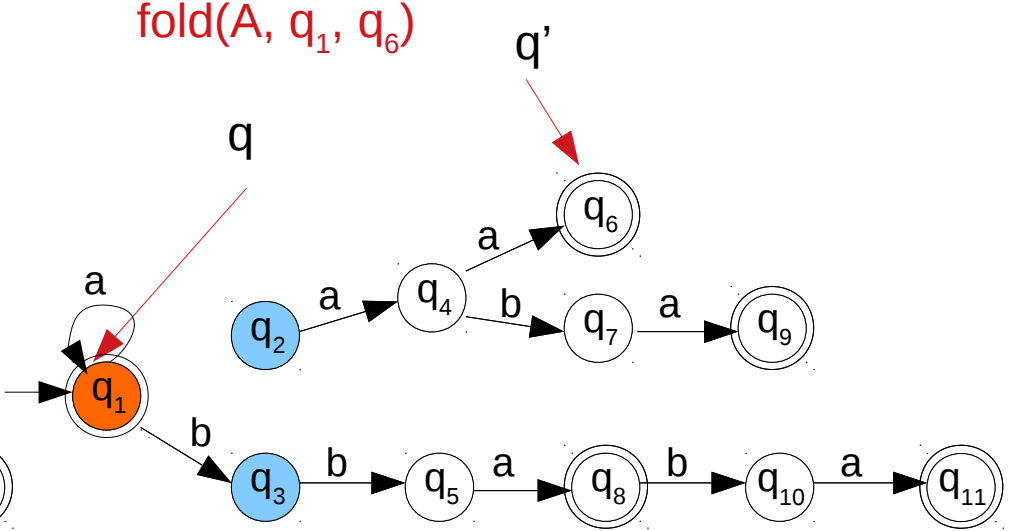
merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se δ(q',s) está definido
        Se δ(q,s) está definido
            A ← fold (A, δ(q,s), δ(q',s))
        Senão δ(q,s) ← δ(q',s)
retorna A

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$



fold(A, q_1, q_4)
fold(A, q_1, q_6)

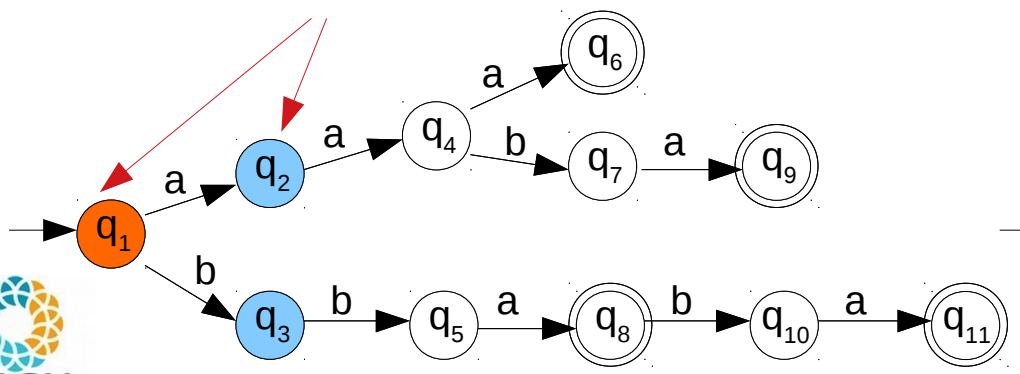


Exemplo - RPNI

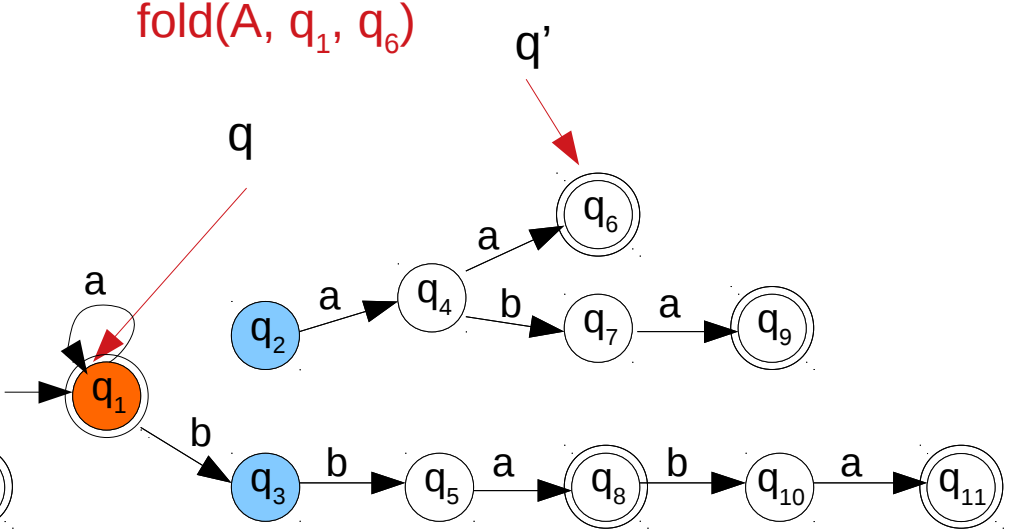
merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se δ(q',s) está definido
        Se δ(q,s) está definido
            A ← fold (A, δ(q,s), δ(q',s))
        Senão δ(q,s) ← δ(q',s)
retorna A

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$



fold(A, q_1, q_4)
fold(A, q_1, q_6)



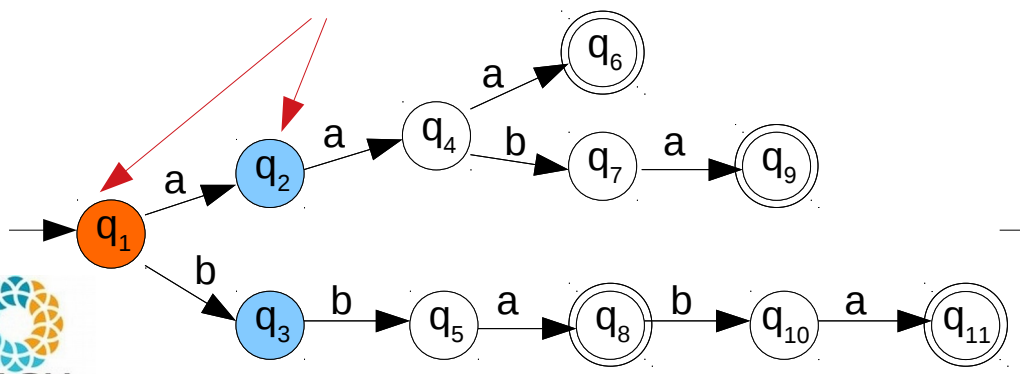
Exemplo - RPNI

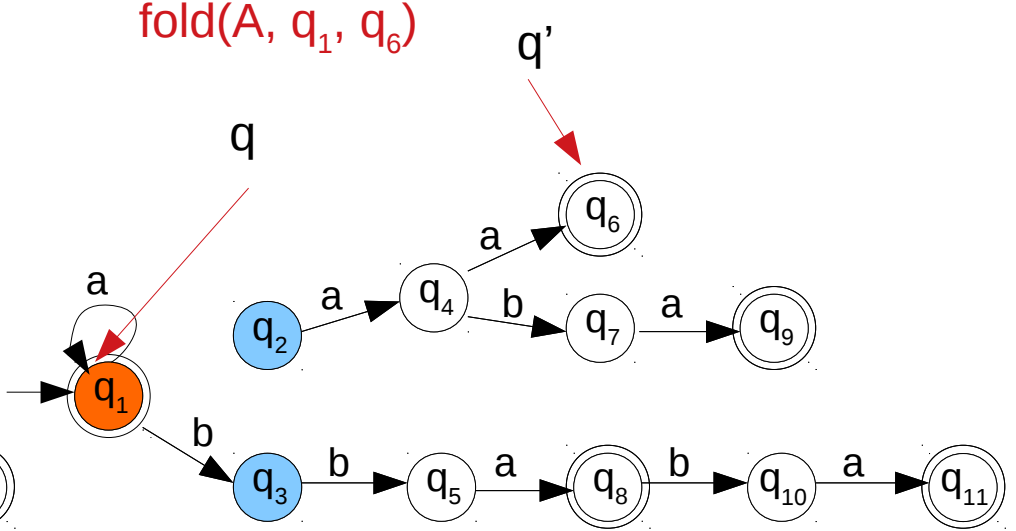
merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se δ(q',s) está definido
        Se δ(q,s) está definido
            A ← fold (A, δ(q,s), δ(q',s))
        Senão δ(q,s) ← δ(q',s)
retorna A

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$

fold(A, q_1, q_4)
fold(A, q_1, q_6)





Exemplo - RPNI

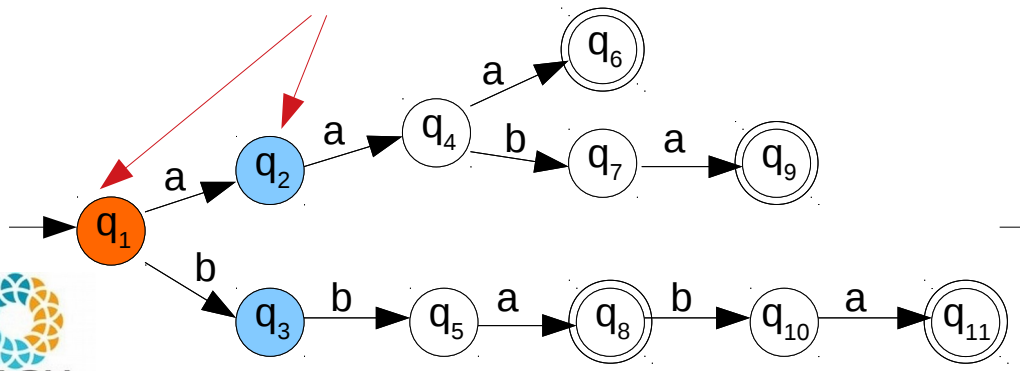
merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

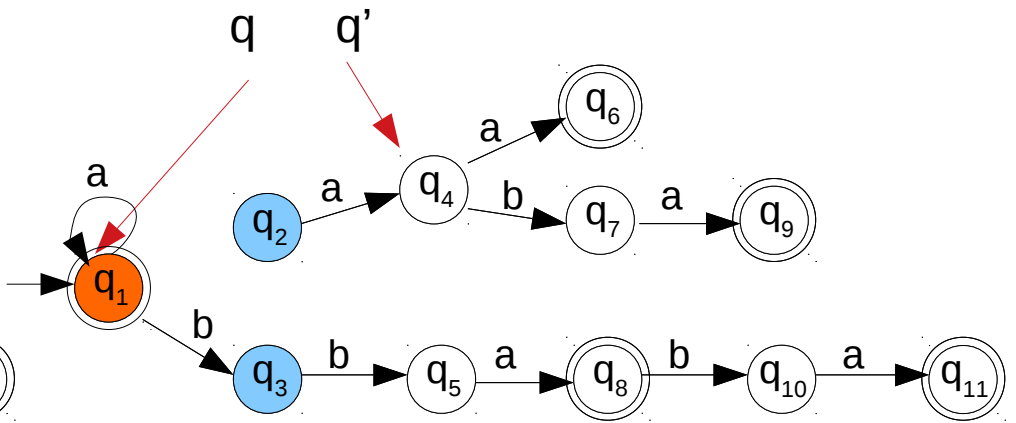
fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se δ(q',s) está definido
        Se δ(q,s) está definido
            A ← fold (A, δ(q,s), δ(q',s))
        Senão δ(q,s) ← δ(q',s)
retorna A

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$

Agora s = b

fold(A, q_1, q_4)





Exemplo - RPNI

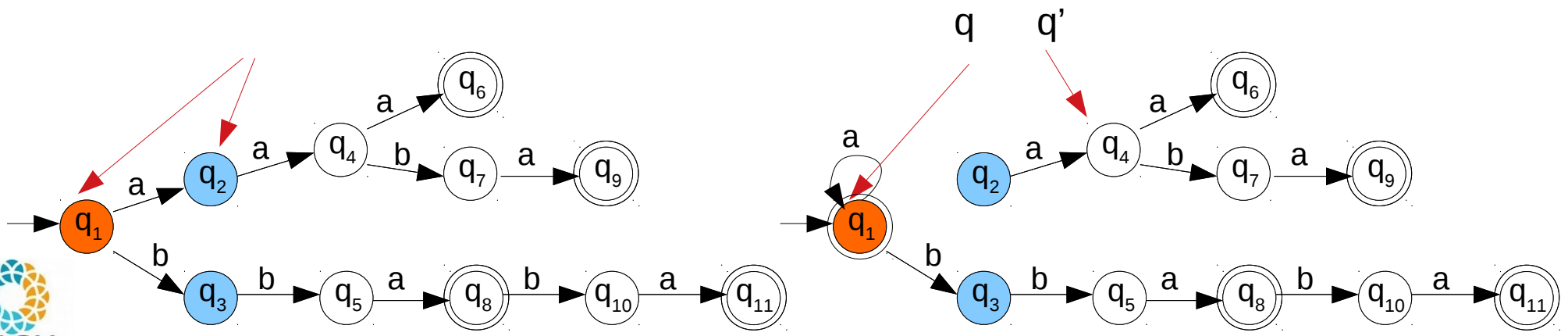
merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se δ(q',s) está definido
        Se δ(q,s) está definido
            A ← fold (A, δ(q,s), δ(q',s))
        Senão δ(q,s) ← δ(q',s)
retorna A

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$

Agora s = b

fold(A, q_1, q_4)
    fold(A, q_3, q_7)



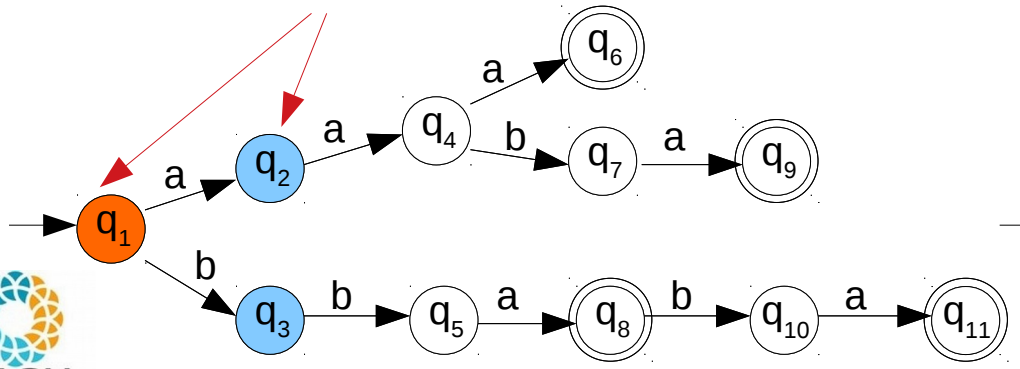
Exemplo - RPNI

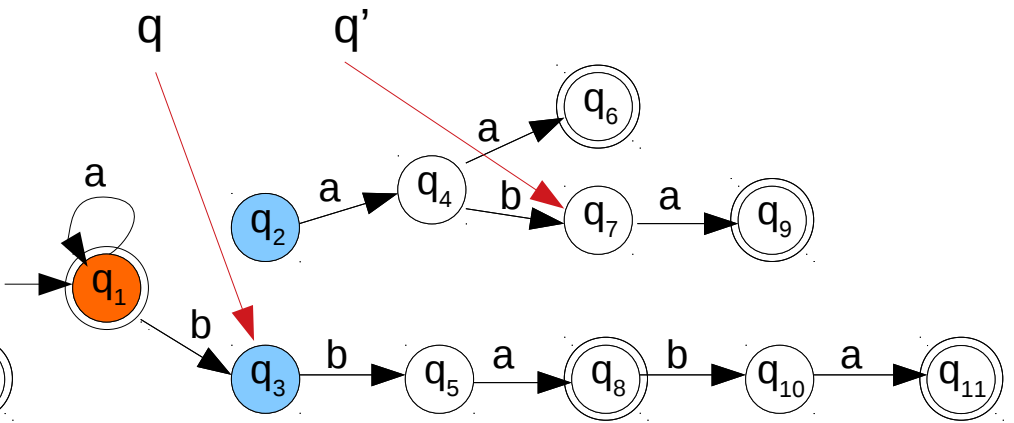
merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se δ(q',s) está definido
        Se δ(q,s) está definido
            A ← fold (A, δ(q,s), δ(q',s))
        Senão δ(q,s) ← δ(q',s)
retorna A

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$

fold(A, q_1, q_4)
    fold(A, q_3, q_7)





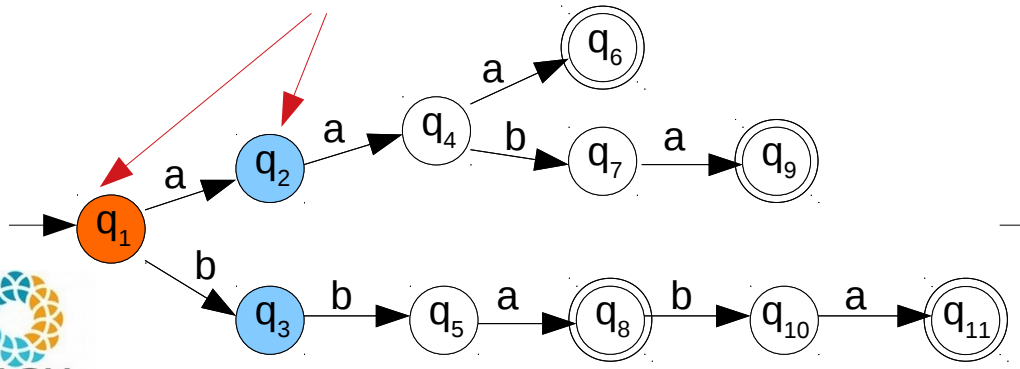
Exemplo - RPNI

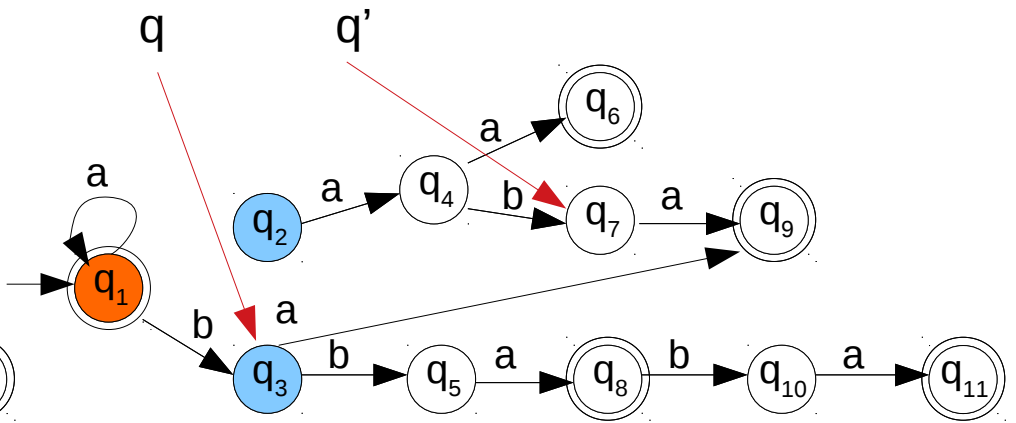
merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se δ(q',s) está definido
        Se δ(q,s) está definido
            A ← fold (A, δ(q,s), δ(q',s))
        Senão δ(q,s) ← δ(q',s)
retorna A

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$

fold(A, q_1, q_4)
    fold(A, q_3, q_7)





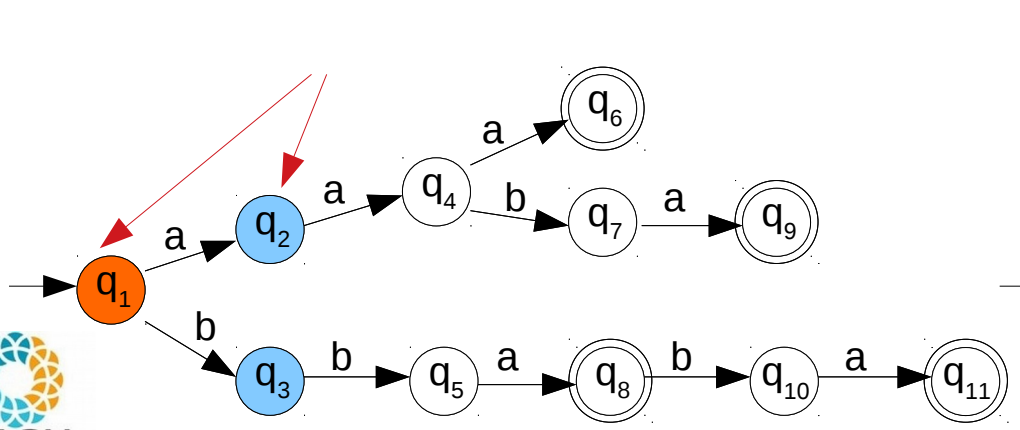
Exemplo - RPNI

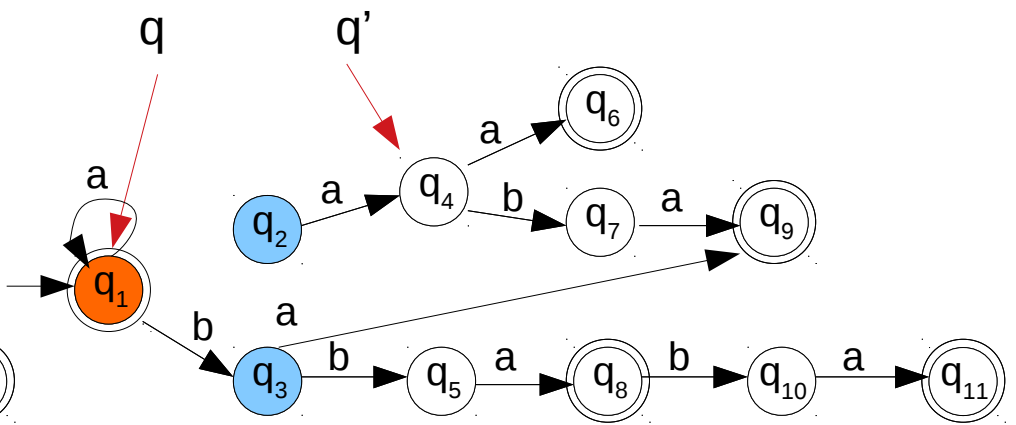
merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se δ(q',s) está definido
        Se δ(q,s) está definido
            A ← fold (A, δ(q,s), δ(q',s))
        Senão δ(q,s) ← δ(q',s)
retorna A

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$

fold(A, q_1, q_4)



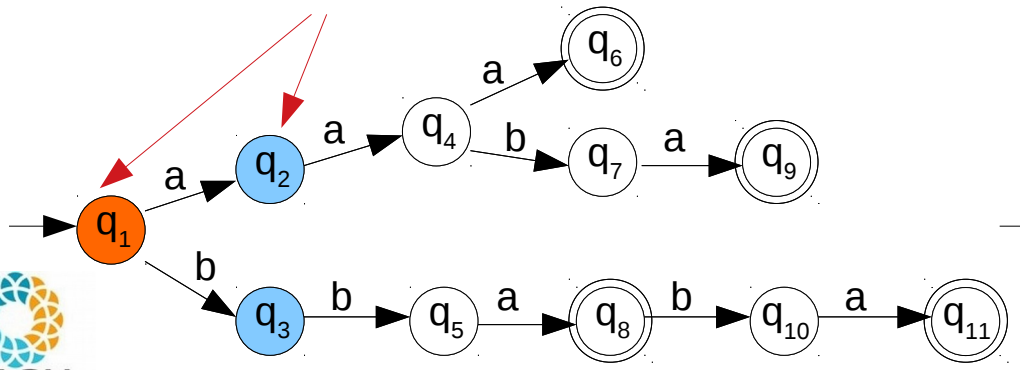


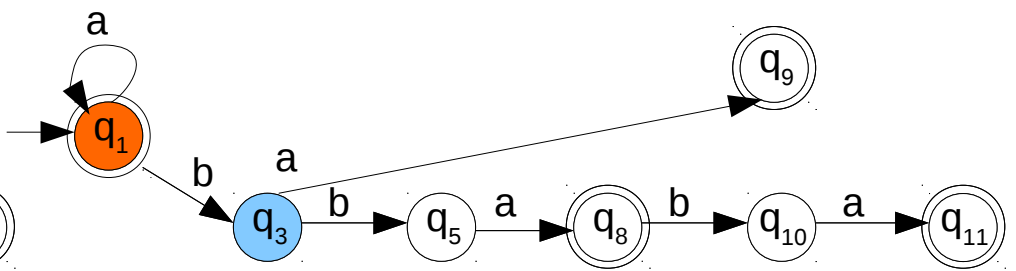
Exemplo - RPNI

merge (A, q, q')
para todo par (q_i, s) tal que $\delta(q_i,s) = q'$
    $\delta(q_i,s) \leftarrow q$
retorna ***fold***(A, q, q')

fold (A, q, q')
Se q' ϵ F_A $F_A \leftarrow F_A \cup \{q\}$
para todo s ϵ Σ
    Se δ(q',s) está definido
        Se δ(q,s) está definido
            A ← fold (A, δ(q,s), δ(q',s))
        Senão δ(q,s) ← δ(q',s)
retorna A

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$





Exemplo - RPNI

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

A ← ***merge***(A, q_r, q_b)

estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

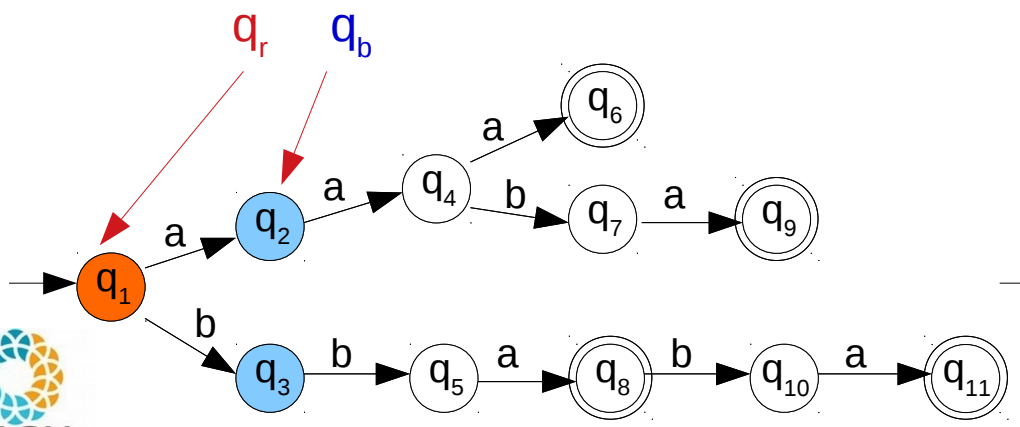
senão ***promove***(A, q_b)

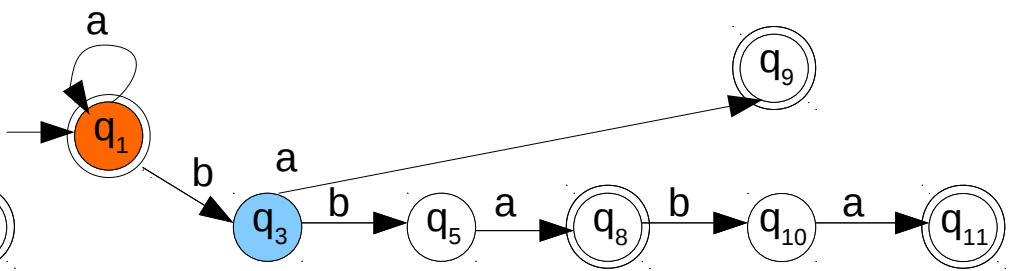
Para cada q

Se q é o estado final de uma cadeia de S^- então $F_R \leftarrow F_R \cup \{q\}$

$S^+ = \{aaa, aaba, bba, bbaba\}$

$S^- = \{a, bb, aab, aba\}$





Exemplo - RPNI

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

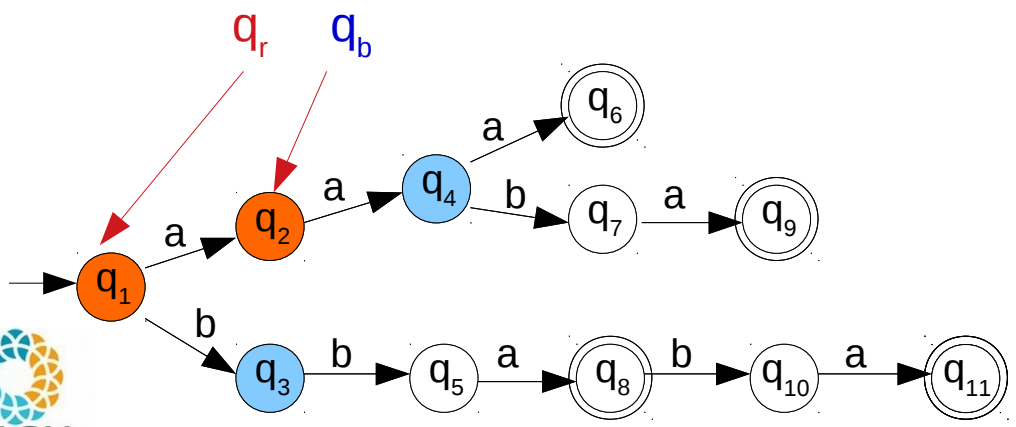
A ← ***merge***(A, q_r, q_b)

estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

senão ***promove***(A, q_b)

Para cada q

Se q é o estado final de uma cadeia de S^- então F_R ← F_R U {q}



Exemplo - RPNI

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

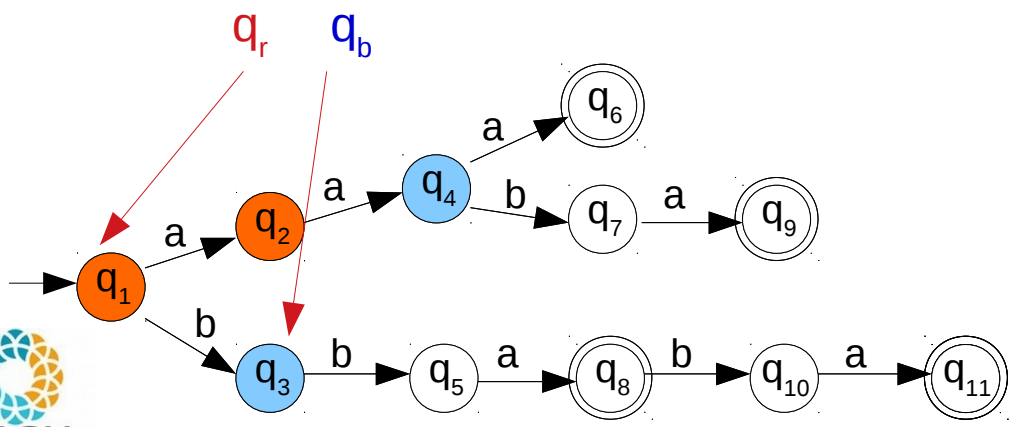
A ← ***merge***(A, q_r, q_b)

estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

senão ***promove***(A, q_b)

Para cada q

Se q é o estado final de uma cadeia de S^- então F_R ← F_R U {q}



Exemplo - RPNI

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

A ← ***merge***(A, q_r, q_b)

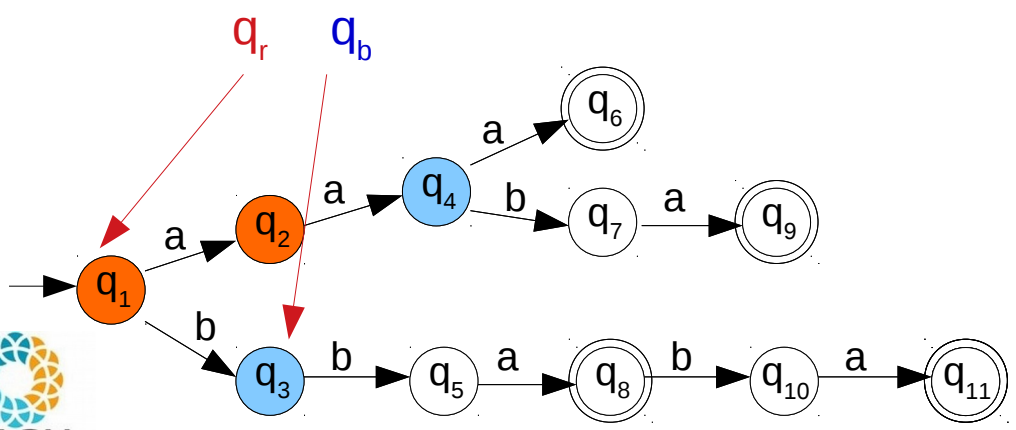
estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

senão ***promove***(A, q_b)

Para cada q

Se q é o estado final de uma cadeia de S^- então F_R ← F_R U {q}

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$



Após vários passos….
(exercício...)

Exemplo - RPNI

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

A ← ***merge***(A, q_r, q_b)

estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

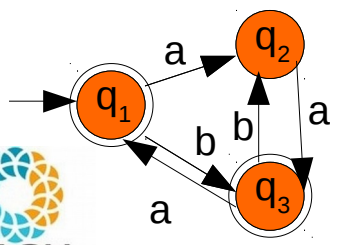
senão ***promove***(A, q_b)

Para cada q

Se q é o estado final de uma cadeia de S^- então F_R ← F_R U {q}

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$

Após vários passos….
(exercício...)



Exemplo - RPNI

Entrada: S^+, S^-

A ← Constrói PTA a partir de S^+

q_0 torna-se vermelho

Todo q tal que $\delta(q_0, s) = q$ para todo s ϵ Σ torna-se azul

enquanto tiver um estado azul

escolha(q_b azul)

se existe q_r que seja ***compatível***(***merge***(A, q_r, q_b), S^-)

A ← ***merge***(A, q_r, q_b)

estados brancos que sejam o próximo estado de estados vermelhos tornam-se azuis

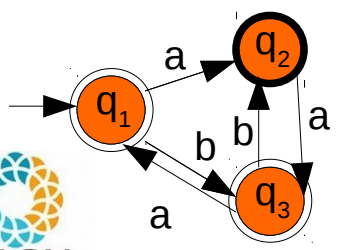
senão ***promove***(A, q_b)

Para cada q

Se q é o estado final de uma cadeia de S^- então $F_R \leftarrow F_R \cup \{q\}$

$S^+ = \{aaa, aaba, bba, bbaba\}$
$S^- = \{a, bb, aab, aba\}$

Após vários passos….
(exercício...)



LAPFA

E autômatos probabilísticos?

- Há também algoritmos
- Na ausência de amostra negativa, utilizam a frequência das cadeias da amostra para controlar a generalização
- Ex:
  - **LAPFA** (RON et al, 1998)
  - Alergia (CARRASCO & ONCINA, 1994)
  - Amnesia (SINGER et al, 1996)

LAPFA

- *Learn Acyclic Probabilistic Finite Automata*
- Idealizado para linguagens finitas, compostas por sequências curtas (AFD acíclico)

AFD estocástico acíclico de níveis

Definição *Define-se um AFDE acíclico (AFDEA) como uma 7-upla* $(Q, q_0, q_f, \Sigma, \zeta, \tau, \gamma)$, *em que:*

- Q *é um conjunto finito de estados;*
- $q_0 \in Q$ *é o estado inicial;*
- $q_f \notin Q$ *é o estado final;*
- Σ *é um alfabeto finito;*
- $\zeta \notin \Sigma$ *é o símbolo final (que representa o final de uma sequência);*
- $\tau : Q \times \Sigma \bigcup \{\zeta\} \rightarrow Q \bigcup \{q_f\}$ *é a função de transição;*
- $\gamma : Q \times \Sigma \bigcup \{\zeta\} \rightarrow [0, 1]$ *é a função probabilística de escolha do próximo símbolo.*

Fonte: Guilherme Miura Lavezzo (2020).

AFD estocástico acíclico de níveis

Definição *Define-se um AFDE acíclico (AFDEA) como uma 7-upla* $(Q, q_0, q_f, \Sigma, \zeta, \tau, \gamma)$, *em que:*

- Q *é um conjunto finito de estados;*
- $q_0 \in Q$ *é o estado inicial;*
- $q_f \notin Q$ *é o estado final;*
- Σ *é um alfabeto finito;*
- $\zeta \notin \Sigma$ *é o símbolo final (que represen*
- $\tau : Q \times \Sigma \bigcup \{\zeta\} \rightarrow Q \bigcup \{q_f\}$ *é a funç*
- $\gamma : Q \times \Sigma \bigcup \{\zeta\} \rightarrow [0, 1]$ *é a função p*

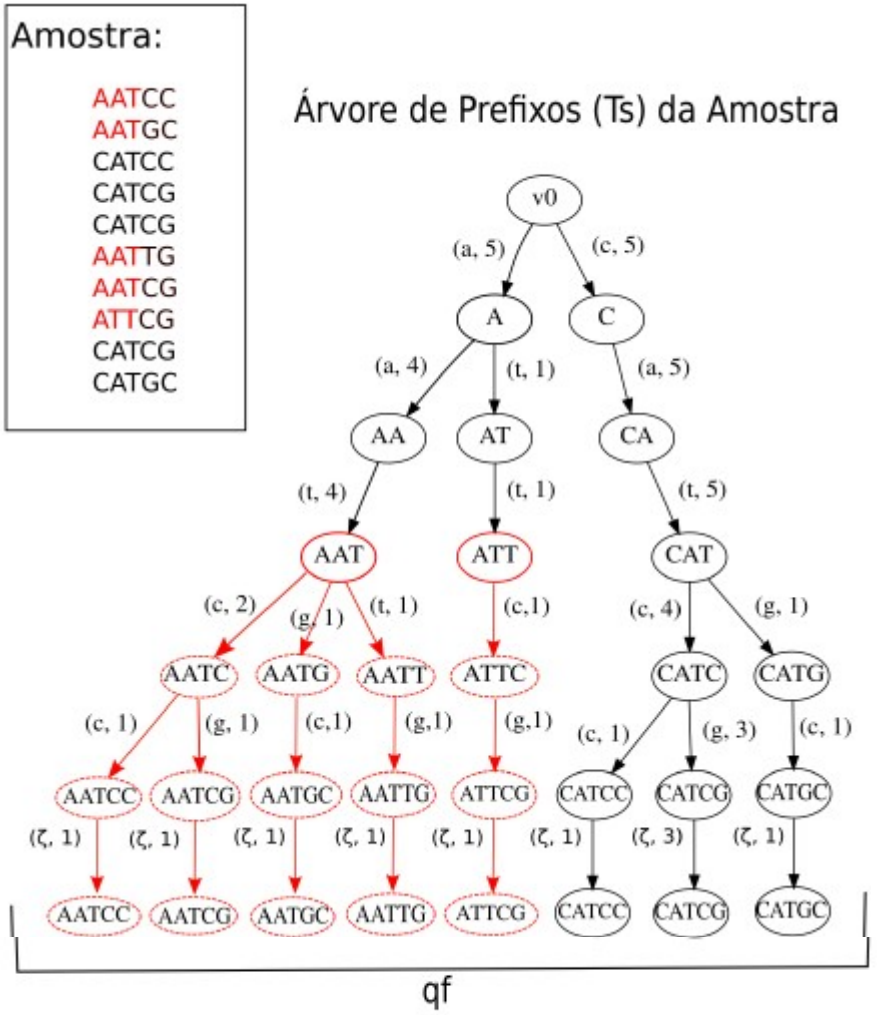
Fonte: Guilherme Miura Lavezzo (2020).

As funções devem satisfazer as seguintes propriedades:

- *para todo* $q \in Q$, $\sum_{\sigma \in \Sigma \bigcup \{\zeta\}} \gamma(q, \sigma) = 1$;
- *a função de transição* τ *só poderá não ser definida para estados* q *e símbolos* σ *em que* $\gamma(q, \sigma) = 0$;
- *para todo* $q \in Q$, $\tau(q, \zeta) = q_f$ *e* $\gamma(q, \zeta) > 0$;
- *o estado* q_f *pode ser alcançado por qualquer estado* $q \in Q$ *que possa ser alcançado a partir de* q_0;
- *a função de transição* τ *só pode ser definida entre dois estados* q *e* r $(r \neq q_f)$ *se* q *for um estado do nível* d *e* r *for um estado do nível* $d + 1$, *para* $d = 0, ..., L$, *sendo 0 o nível da raiz (na qual encontra-se o estado* q_0*) e* L *o penúltimo*[8] *nível do autômato (sendo esta última propriedade a que torna o AFDE acíclico).*

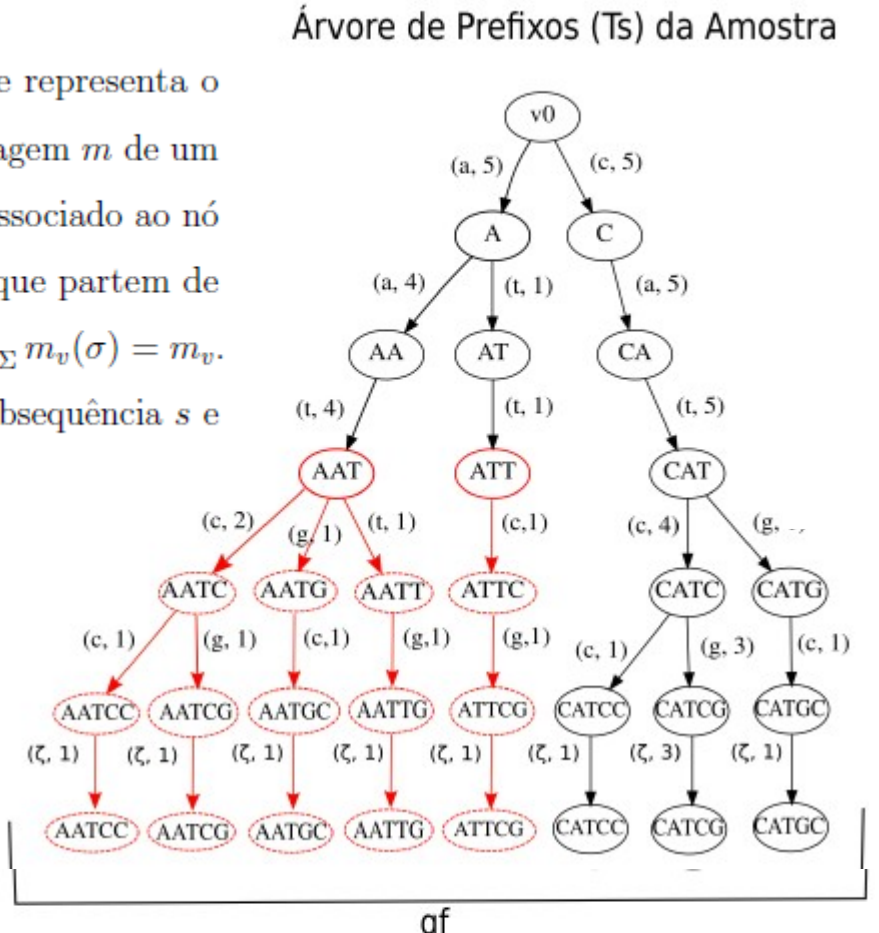
Exemplo

Note que esse é o PTA ajustado para a definição anterior (estado final q_f e símbolo de fim de cadeia)



Mais definições

Além do rótulo σ, cada aresta contém também uma contagem que representa o número de sequências em S que passam por aquela trajetória. Isto é, a contagem m de um nó v, m_v, representa o número de sequências em S que possuem o prefixo associado ao nó v. De forma semelhante, $m_v(\sigma)$ representa o número de sequências em S que partem de v sob a aresta rotulada por σ. Tem-se então que vale a propriedade : $\sum_{\sigma \in \Sigma} m_v(\sigma) = m_v$. Além disso, $m_v(s)$ representa o número de sequências de S que contêm a subsequência s e que s é gerada por M a partir do nó v.



Algoritmo LAPFA

Parâmetros: m_0, μ, γ_{min}

Constrói PTA como citado a partir de S^+

Para cada nível do autômato

Para cada par de estados i e j deste nível

Se $m_i > m_0$ e $m_j > m_0$ e ***similar***(i, 1, j, 1)

junta(i, j) //e todos os pares de estados que eles alcançam

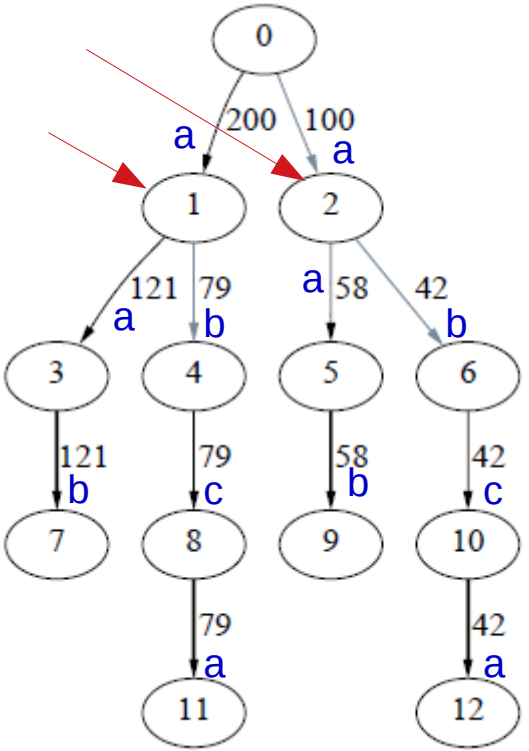
adiciona_folga

calcula_probabilidade

Algoritmo LAPFA

similar(i, p_i, j, p_j)

se $|p_i - p_j| \geq \mu/2$ então
  devolve `falso`
senão se $p_i < \mu/2$ e $p_j < \mu/2$ então
  devolve `verdadeiro`
senão
  para todo $\sigma \in \Sigma \cup \zeta$ faça
    $p'_i \leftarrow p_i m_i(\sigma)/m_i$
    $p'_j \leftarrow p_j m_j(\sigma)/m_j$
    se $\text{similar}(\tau(i,\sigma), p'_i, \tau(j,\sigma), p'_j)$ = falso
      devolve `falso`
    fim se
  fim para
fim se
devolve `verdadeiro` {chamadas recursivas acabaram, logo são similares}

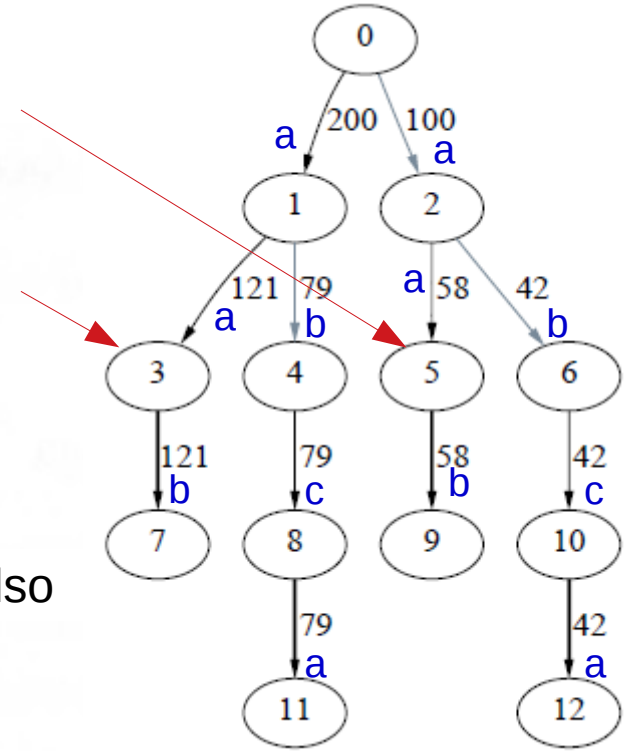


Ex: μ = 0,1
similar (1, 1, 2, 1)
$p_1 = 1$, $p_2 = 1$
Para σ = a:
$p'_1 = 1x(121/200)=0.605$
$p'_2 = 1x(58/100)=0.58$

Algoritmo LAPFA

similar(i, p_i, j, p_j)

se $|p_i - p_j| \geq \mu/2$ então
  devolve `falso`
senão se $p_i < \mu/2$ e $p_j < \mu/2$ então
  devolve `verdadeiro`
senão
  para todo $\sigma \in \Sigma \cup \zeta$ faça
    $p'_i \leftarrow p_i m_i(\sigma)/m_i$
    $p'_j \leftarrow p_j m_j(\sigma)/m_j$
    se $\text{similar}(\tau(i,\sigma), p'_i, \tau(j,\sigma), p'_j)$ = falso
      devolve `falso`
    fim se
  fim para
fim se
devolve `verdadeiro` {chamadas recursivas acabaram, logo são similares}

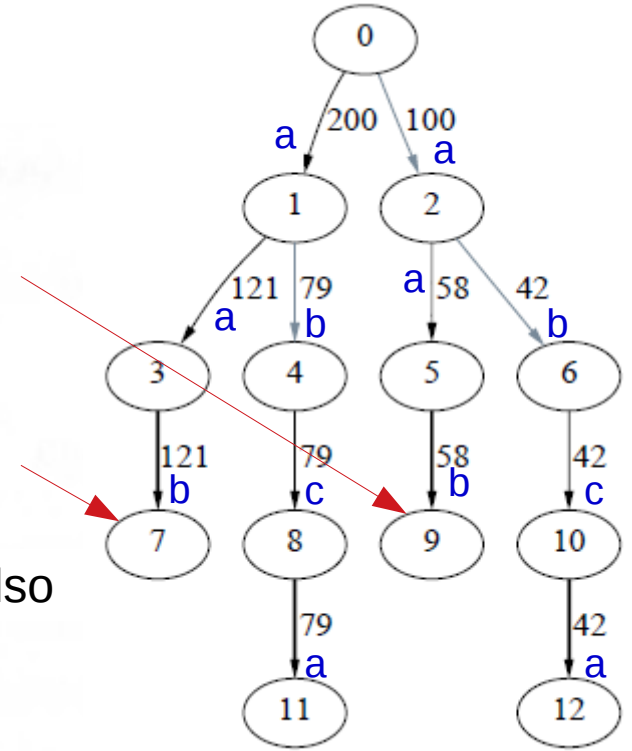


Ex: μ = 0,1
similar (1, 1, 2, 1)
$p_1 = 1$, $p_2 = 1$
Para σ = a:
$p'_1 = 1x(121/200)=0.605$
$p'_2 = 1x(58/100)=0.58$
similar(3,0.605,5,0.58)
Para σ = b:
$p'_3=0.605x(121/121)=0.605$
$p'_5=0.58x(58/58)=0.58$

Algoritmo LAPFA

similar(i, p_i, j, p_j)

se $|p_i - p_j| \geq \mu/2$ então
  devolve `falso`
senão se $p_i < \mu/2$ e $p_j < \mu/2$ então
  devolve `verdadeiro`
senão
  para todo $\sigma \in \Sigma \cup \zeta$ faça
    $p'_i \leftarrow p_i m_i(\sigma)/m_i$
    $p'_j \leftarrow p_j m_j(\sigma)/m_j$
    se $\text{similar}(\tau(i,\sigma), p'_i, \tau(j,\sigma), p'_j)$ = falso
      devolve `falso`
    fim se
  fim para
fim se
devolve `verdadeiro` {chamadas recursivas acabaram, logo são similares}

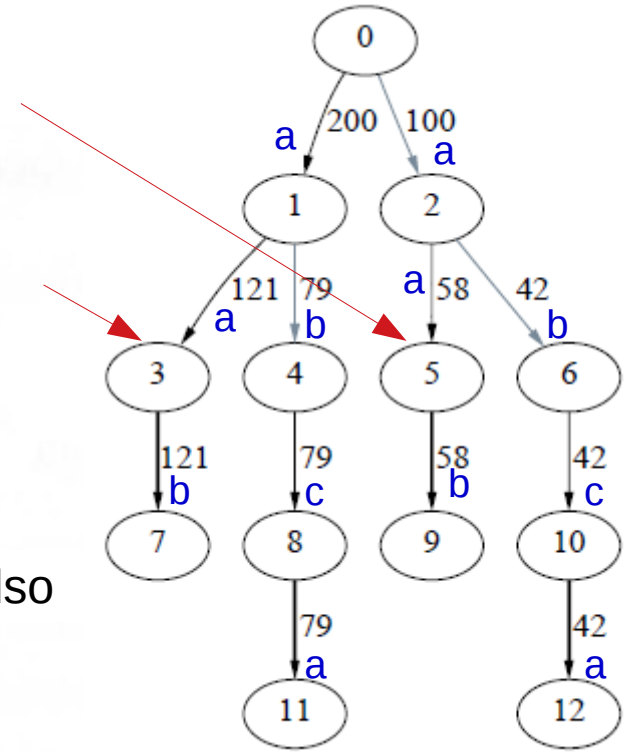


Ex: μ = 0,1
similar (1, 1, 2, 1)
$p_1 = 1$, $p_2 = 1$
Para σ = a:
$p'_1 = 1x(121/200)=0.605$
$p'_2 = 1x(58/100)=0.58$
similar(3,0.605,5,0.58)
Para σ = b:
$p'_3=0.605x(121/121)=0.605$
$p'_5=0.58x(58/58)=0.58$
similar(7,0.605,9,0.58)
retorna verdadeiro

Algoritmo LAPFA

similar(i, p_i, j, p_j)

se $|p_i - p_j| \geq \mu/2$ então
  devolve falso
senão se $p_i < \mu/2$ e $p_j < \mu/2$ então
  devolve verdadeiro
senão
  para todo $\sigma \in \Sigma \cup \zeta$ faça
    $p'_i \leftarrow p_i m_i(\sigma)/m_i$
    $p'_j \leftarrow p_j m_j(\sigma)/m_j$
    se similar$(\tau(i,\sigma), p'_i, \tau(j,\sigma), p'_j)$ = falso
      devolve falso
    fim se
  fim para
fim se
devolve verdadeiro {chamadas recursivas acabaram, logo são similares}

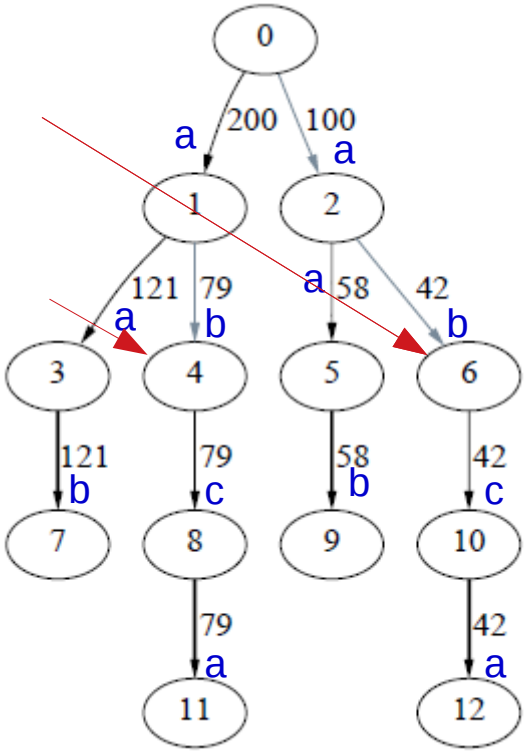


Ex: μ = 0,1
similar (1, 1, 2, 1)
$p_1 = 1$, $p_2 = 1$
Para σ = a:
$p'_1 = 1x(121/200)=0.605$
$p'_2 = 1x(58/100)=0.58$
similar(3,0.605,5,0.58)
Para σ = b:
$p'_3=0.605x(121/121)=0.605$
$p'_5=0.58x(58/58)=0.58$
similar(7,0.605,9,0.58)
retorna verdadeiro
retorna verdadeiro

Algoritmo LAPFA

similar(i, p_i, j, p_j)

se $|p_i - p_j| \geq \mu/2$ então
  devolve falso
senão se $p_i < \mu/2$ e $p_j < \mu/2$ então
  devolve verdadeiro
senão
  para todo $\sigma \in \Sigma \cup \zeta$ faça
    $p'_i \leftarrow p_i m_i(\sigma)/m_i$
    $p'_j \leftarrow p_j m_j(\sigma)/m_j$
    se similar$(\tau(i,\sigma), p'_i, \tau(j,\sigma), p'_j)$ = falso
      devolve falso
    fim se
  fim para
fim se
devolve verdadeiro {chamadas recursivas acabaram, logo são similares}

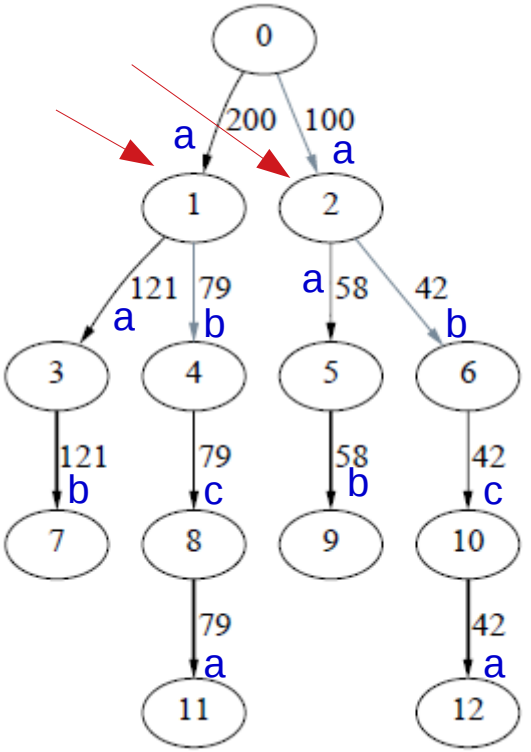


Ex: μ = 0,1
similar (1, 1, 2, 1)
$p_1 = 1$, $p_2 = 1$
Para σ = a:
$p'_1 = 1x(121/200)=0.605$
$p'_2 = 1x(58/100)=0.58$
similar(3,0.605,5,0.58)
Para σ = b:
$p'_3=0.605x(121/121)=0.605$
$p'_5=0.58x(58/58)=0.58$
similar(7,0.605,9,0.58)
retorna verdadeiro
retorna verdadeiro
Para σ = b:
$p'_1 = 1x(79/200)=0.395$
$p'_2 = 1x(42/100)=0.42$
similar(4,0.395,6,0.42)
...

Algoritmo LAPFA

similar(i, p_i, j, p_j)

se $|p_i - p_j| \geq \mu/2$ então
  devolve falso
senão se $p_i < \mu/2$ e $p_j < \mu/2$ então
  devolve verdadeiro
senão
  para todo $\sigma \in \Sigma \cup \zeta$ faça
    $p'_i \leftarrow p_i m_i(\sigma)/m_i$
    $p'_j \leftarrow p_j m_j(\sigma)/m_j$
    se similar$(\tau(i,\sigma), p'_i, \tau(j,\sigma), p'_j)$ = falso
      devolve falso
    fim se
  fim para
fim se
devolve verdadeiro {chamadas recursivas acabaram, logo são similares}

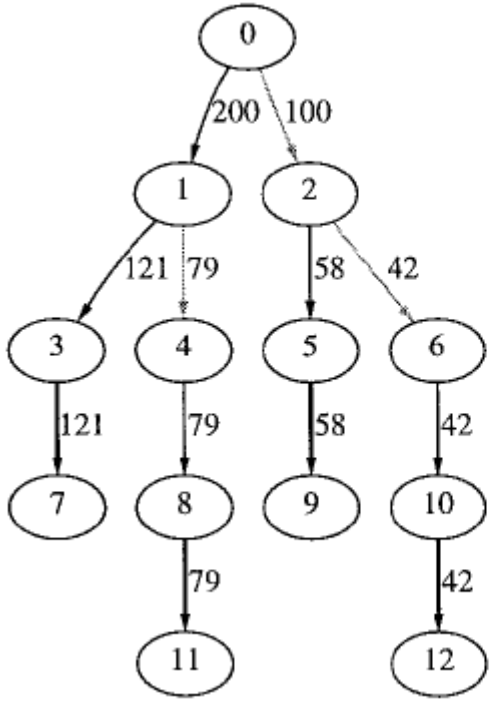


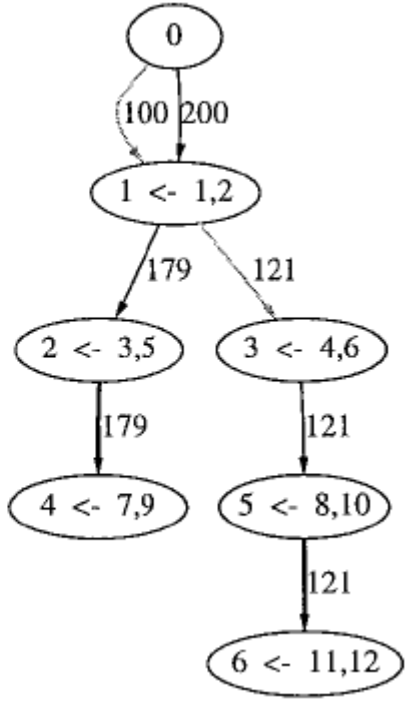
Ex: μ = 0,1
similar (1, 1, 2, 1)
$p_1 = 1$, $p_2 = 1$
Para σ = a:
$p'_1 = 1x(121/200)=0.605$
$p'_2 = 1x(58/100)=0.58$
similar(3,0.605,5,0.58)
Para σ = b:
$p'_3=0.605x(121/121)=0.605$
$p'_5=0.58x(58/58)=0.58$
similar(7,0.605,9,0.58)
retorna verdadeiro
retorna verdadeiro
Para σ = b:
$p'_1 = 1x(79/200)=0.395$
$p'_2 = 1x(42/100)=0.42$
similar(4,0.395,6,0.42)
...
retorna verdadeiro

Algoritmo LAPFA

similares(i, p_i, j, p_j)

se $|p_i - p_j| \geq \mu/2$ então
  devolve `falso`
senão se $p_i < \mu/2$ e $p_j < \mu/2$ então
  devolve `verdadeiro`
senão
  para todo $\sigma \in \Sigma \cup \zeta$ faça
    $p'_i \leftarrow p_i m_i(\sigma)/m_i$
    $p'_j \leftarrow p_j m_j(\sigma)/m_j$
    se similar$(\tau(i,\sigma), p'_i, \tau(j,\sigma), p'_j)$ = falso
      devolve `falso`
    fim se
  fim para
fim se
devolve `verdadeiro` {chamadas recursivas acabaram, logo são similares}



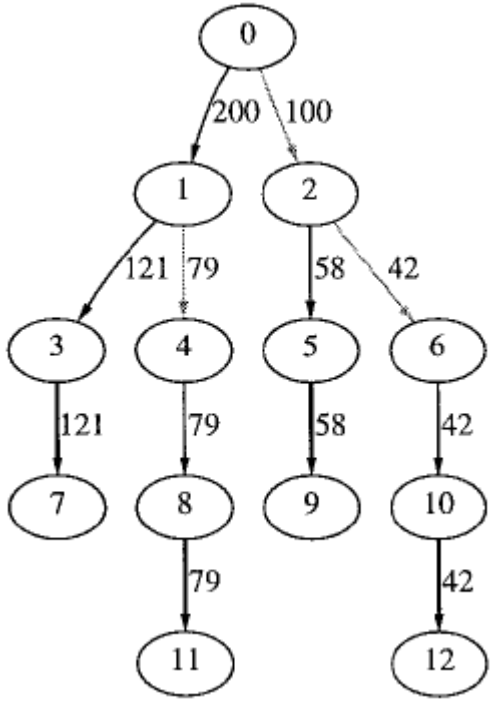


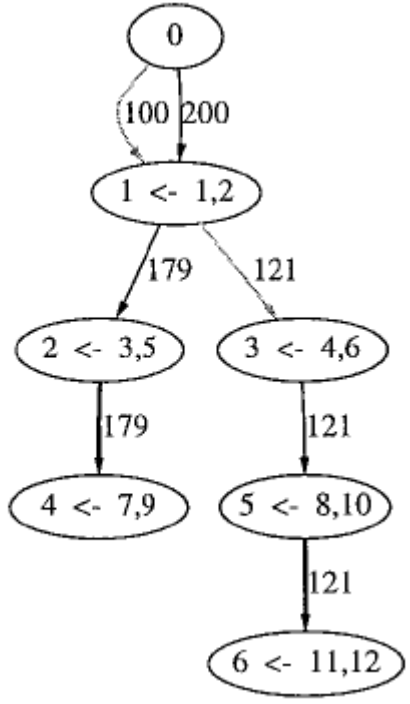
Journal of Computer and System Sciences **56**, 133–152 (1998)
Article No. SS971555

Algoritmo LAPFA

junta(i, j)

para todo estado k e todo $\sigma \in \Sigma \cup \zeta$ tal que $\tau(k, \sigma) = j$ **faça**
  $\tau(k, \sigma) \leftarrow i$
fim para
para todo $\sigma \in \Sigma \cup \zeta$ **faça**
  se $m_i(\sigma) = 0$ e $m_j(\sigma) > 0$ **então**
    $\tau(i, \sigma) \leftarrow \tau(j, \sigma)$
  fim se
  se $m_i(\sigma) > 0$ e $m_j(\sigma) > 0$ **então**
    junta$(\tau(i, \sigma), \tau(j, \sigma), A)$
  fim se
  $m_i(\sigma) \leftarrow m_i(\sigma) + m_j(\sigma)$
fim para





Journal of Computer and System Sciences **56**, 133–152 (1998)
Article No. SS971555

Algoritmo LAPFA

Parâmetros: m_0, μ, γ_{min}

Constrói PTA como citado a partir de S^+

Para cada nível do autômato

Para cada par de estados i e j deste nível

Se $m_i > 0$ e $m_j > 0$ e ***similares***(i, 1, j, 1)

junta(i, j) //e todos os pares de estados que eles alcançam

adiciona_folga 

calcula_probabilidade

adiciona_folga

junte todos estados finais de A num único estado q_f, o qual pertence ao nível D
para $d \leftarrow 1$ **até** $D-1$ **faça**
  junte todos os estados k no nível d em que $m_k < m_0$ no estado $pequeno(d)$
para $d \leftarrow 0$ **até** $D-1$ **faça**
  para todo estado k no nível d **faça**
    para todo $\sigma \in \Sigma$ **faça**
      se $m_k(\sigma) = 0$ **então**
        $\tau(k, \sigma) \leftarrow pequeno(d+1)$
      se $m_k(\zeta) = 0$ **então**
        $\tau(k, \zeta) \leftarrow q_f$

Algoritmo LAPFA

Parâmetros: m_0, μ, γ_{min}

Constrói PTA como citado a partir de S^+

Para cada nível do autômato

Para cada par de estados i e j deste nível

Se $m_i > 0$ e $m_j > 0$ e ***similares***(i, 1, j, 1)

junta(i, j) //e todos os pares de estados que eles alcançam

adiciona_folga

calcula_probabilidade ←

calcula_probabilidade

para todo estado k em A **faça**
  para todo $\sigma \in \Sigma \cup \zeta$ **faça**
    $\gamma(k, \sigma) \leftarrow (m_k(\sigma)/m_k)(1 - (|\Sigma| + 1)\gamma_{min}) + \gamma_{min}$

Exemplo de resultado

FIG. 4. Synthetic cursive letters created by random walks using the APFA that represents the letter k.

Journal of Computer and System Sciences **56**, 133–152 (1998)
Article No. SS971555

Referências

CARRASCO, R. C. and ONCINA, J. Learning stochastic regular grammars by means of a state merging method. Proceedings of the Second International Colloquium. September 21-23. ICG, Alicante, 139-152, 1994.

ONCINA, J.; GARCIA, P. Inferring regular languages in polynomial update time. In: Pattern recognition and image analysis. World Scientific Publishing, Singapore, 49-61, 1992.

RON, D.; SINGER, Y. and TISHBY, N. The power of amnesia: learning probabilistic automata with variable memory length. Machine Learning 25: 117-150, 1996.

RON, D.; SINGER, Y. and TISHBY, N. On the learnability and usage of acyclic probabilistic finite automata. J. Comput. Syst. Sci. 56: 133-152. 1998