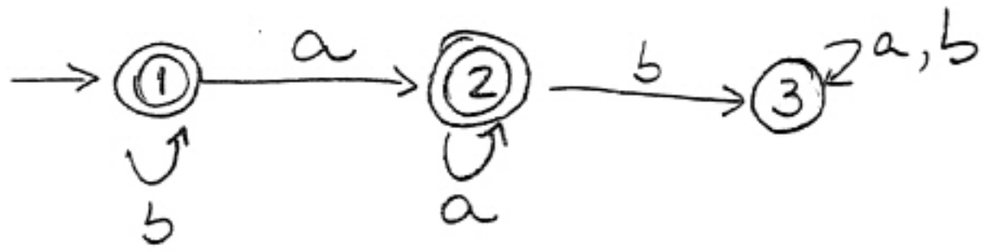
Resolução Lista 1 - ITC

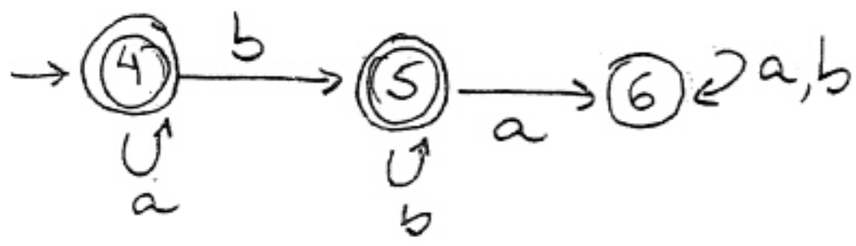
1.5

c) {w| w não contém nem a subcadeia ab, nem ba}

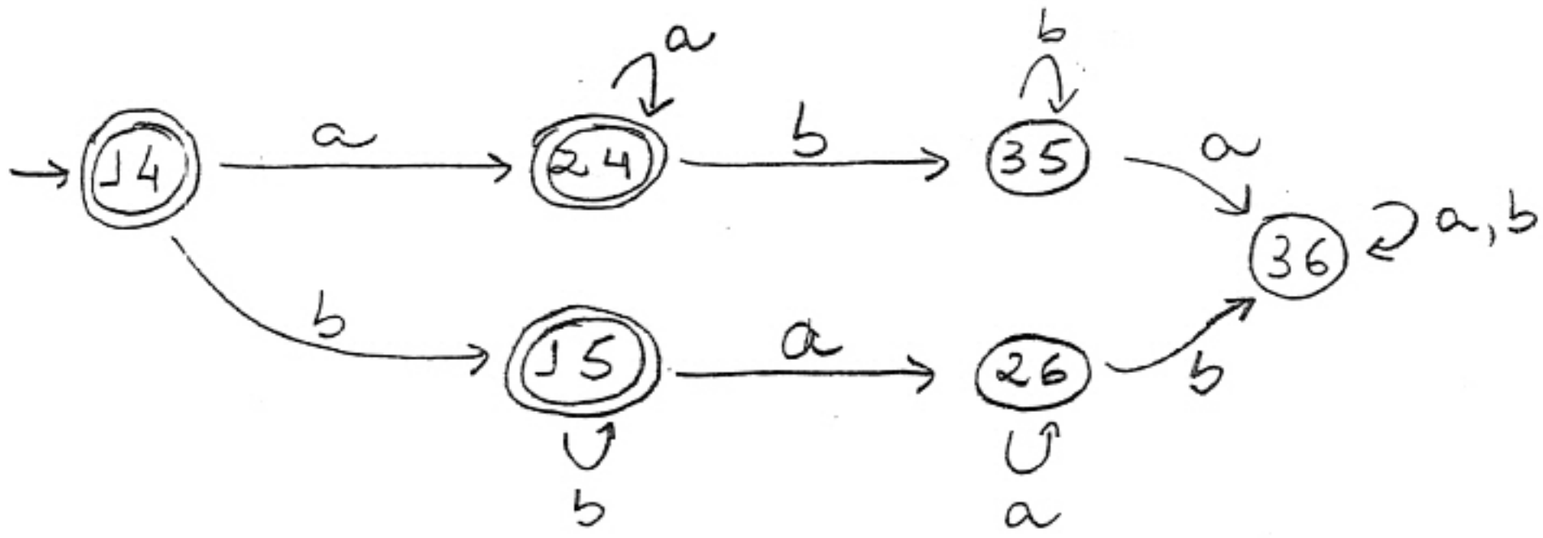
AFD para L_1={ w| w não contém ab }



AFD para L_2={ w| w não contém subcadeia ba }



AFD para $L_1 \cap L_2$:

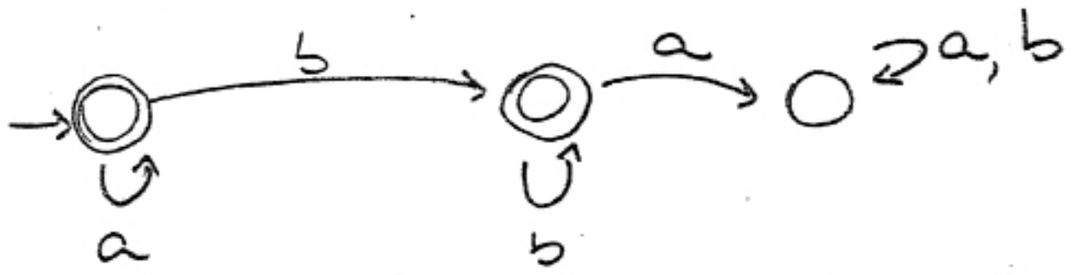


1.5

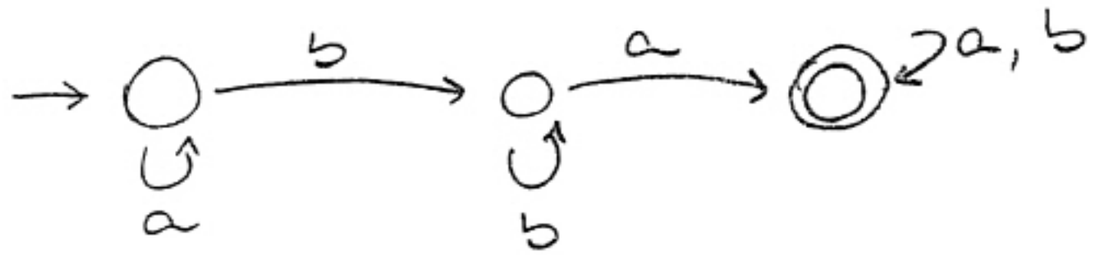
d) {w | w é qualquer cadeia que não está em a^*b^*}

$L_1 = \{w \mid w \text{ está em } a^*b^*\}$

AFD para L_1:



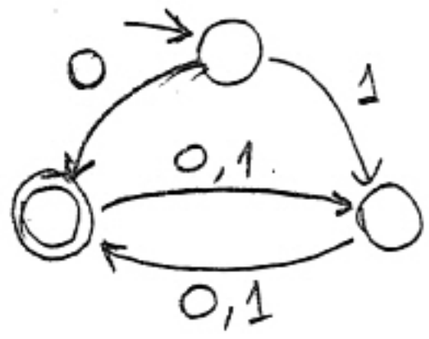
AFD para L_1^c: basta inverter estados de aceitação:



1.6)

e) {w | w começa com 0 e tem comprimento ímpar ou começa com 1 e tem comprimento par}

Podemos construir AFDs para linguagens mais simples e depois construir um AFD a partir dos mais simples; ou podemos construir um AFD diretamente para a linguagem completa. Isso é fácil, uma vez que a paridade é determinada pelo símbolo inicial. Note que como w precisa começar com 0 ou 1, a palavra vazia não deve ser aceita:



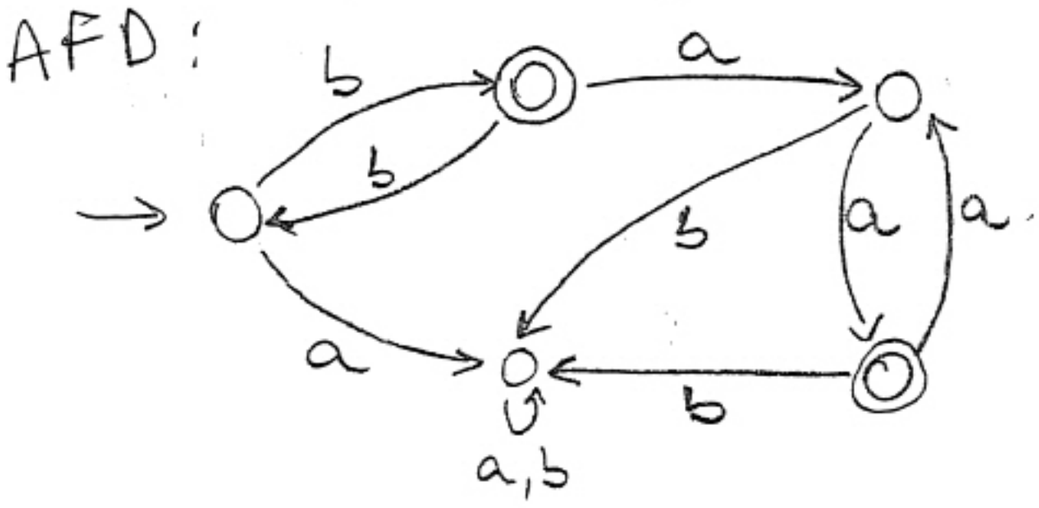
PS 2

1.12) Construir um AFD com 5 estados para

$D = \{w \mid w$ contém um número par de a's e um número ímpar de b's e não contém a subcadeia $ab\}$.

Note que toda palavra deve conter pelo menos um b, e portanto não pode começar com a (pois senão haveria pelo menos uma ocorrência de ab). Também, um a só pode suceder um b se um número ímpar de b's for lido até aquele momento.

AFD:

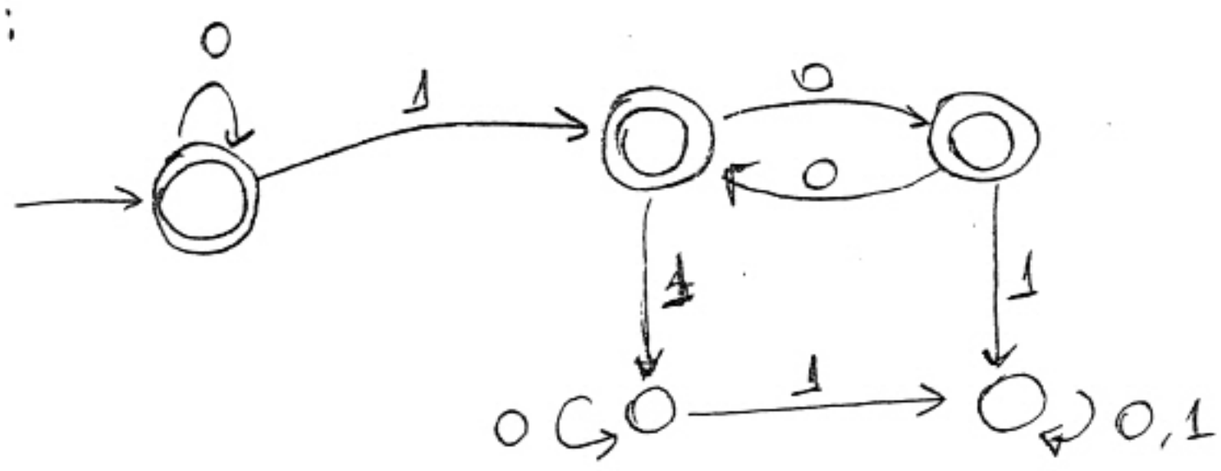


ER:

$b(bb)^*(aa)^*$

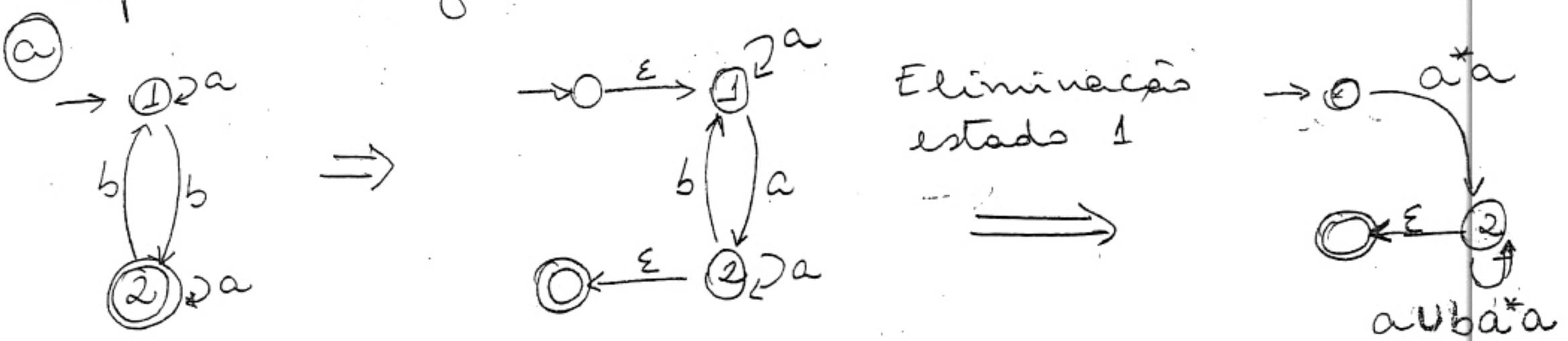
1.13) Construir um AFD de 5 estados para a linguagem $F = \{w \in \{0,1\}^* \mid w$ não contém um par de 1's separados por um número ímpar de 1's$\}$.

Note que um corolário da definição acima é que toda palavra F não conterá mais do que dois 1's, simplificando a construção do AFD:



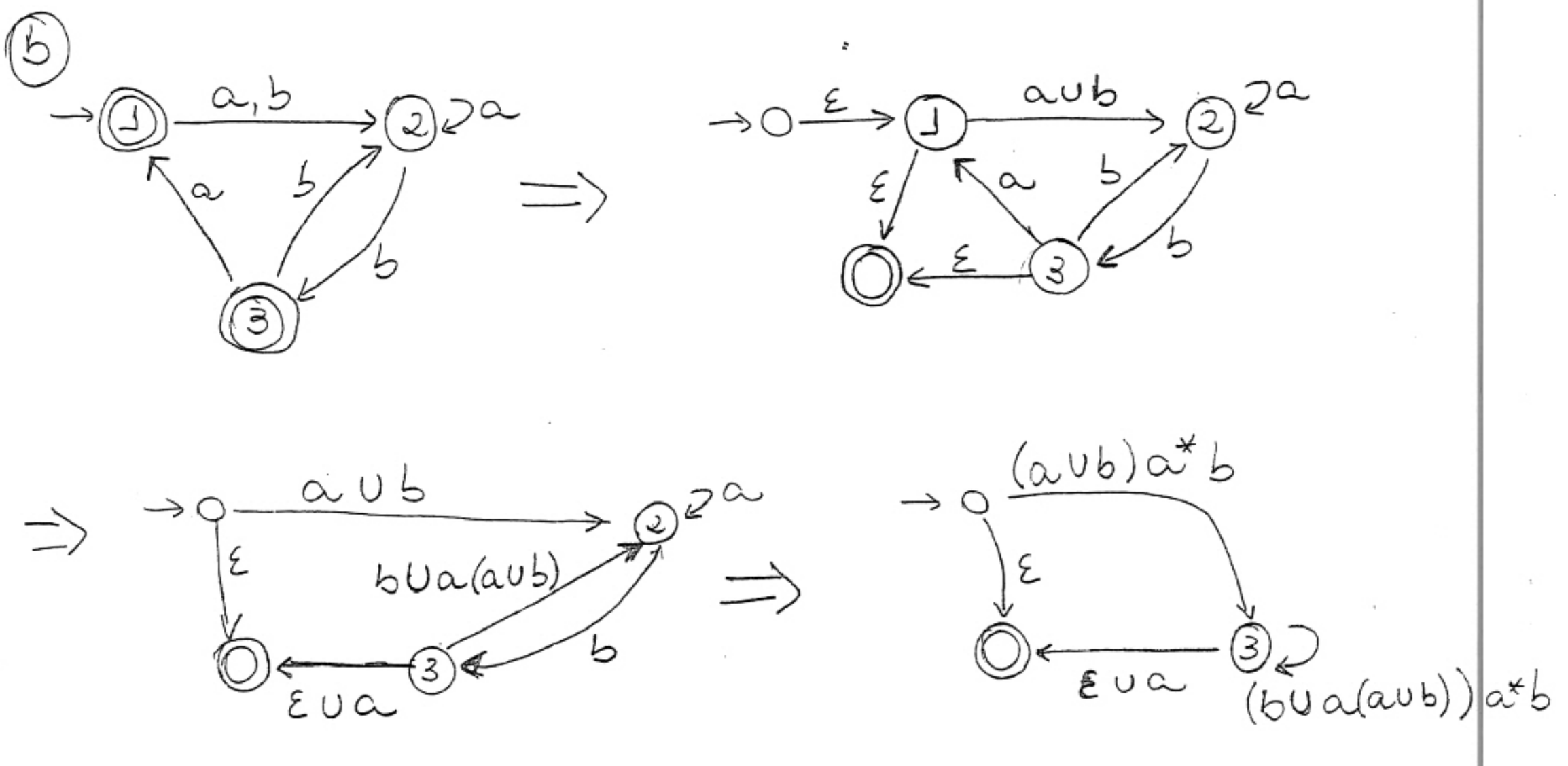
1.21) Conversão de autômatos finitos para expressões regulares:

a)



$\Rightarrow \rightarrow \bigcirc \xrightarrow{a^*a(a \cup ba^*a)} \bigcirc$

b)



$\Rightarrow \rightarrow \bigcirc \xrightarrow{\varepsilon \cup (a \cup b)a^*b\,((b \cup a(a \cup b))a^*b)^*\,(\varepsilon \cup a)} \bigcirc$

construir um AFD M_1' a partir de M_1, simplesmente aplicando o método descrito no enunciado (definindo $F' = Q - F$). Este autômato reconhecerá o complemento de linguagem.

1.31) Se A é regular, A^R também o é.

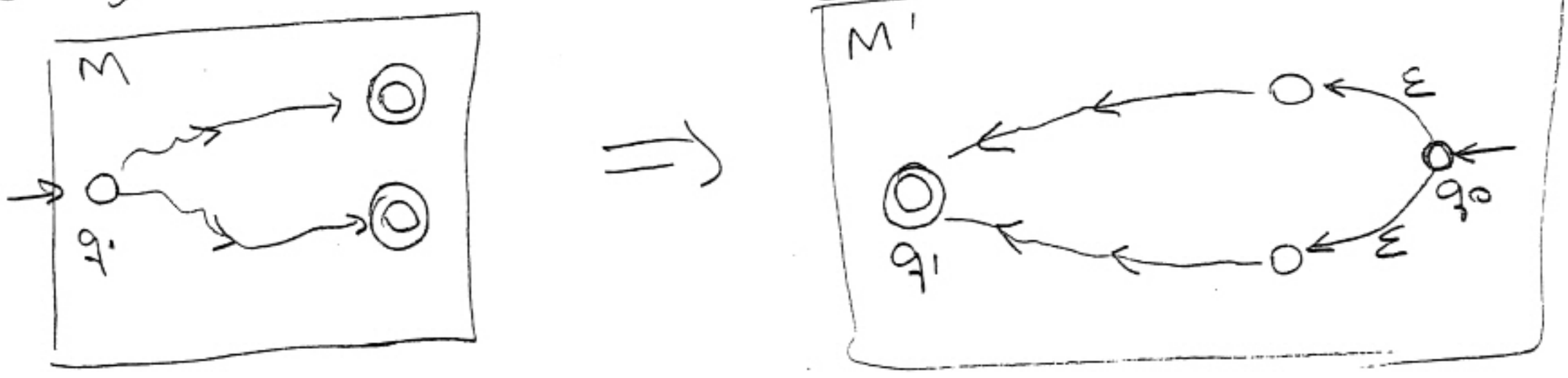
Dem. Seja A uma linguagem regular, e seja $M = (Q, \Sigma, \delta, q_1, F)$ um AFD que reconhece A. Logo, qualquer palavra $W = w_1 w_2 \ldots w_n$ pertencerá a A se e somente for aceita por M, isto é se houver uma sequência de estados $r_0 r_1 r_2 \ldots r_n$ tais que:

$r_0 = q_1$ (estado inicial em M)

$r_i = \delta(r_{i-1}, w_i)$, para $i = 1, \ldots, n$

$r_n \in F$

A idéia é criar um AFN M' invertendo a orientação das transições em δ, tornando o estado q_1 (estado inicial em M) como estado de aceitação em M'. Criaremos em M' um novo estado q_0, que será também o estado inicial. Ligamos q_0 a todos os estados que eram de aceitação em M, com cadeia vazia. A idéia é resumida no diagrama abaixo.

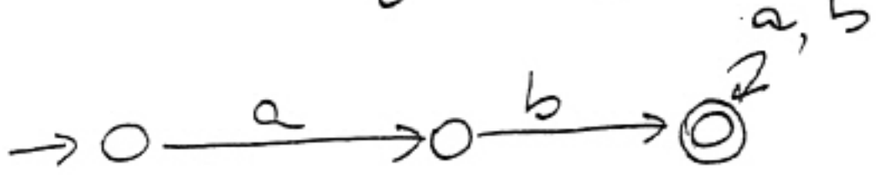


continua → pág 5

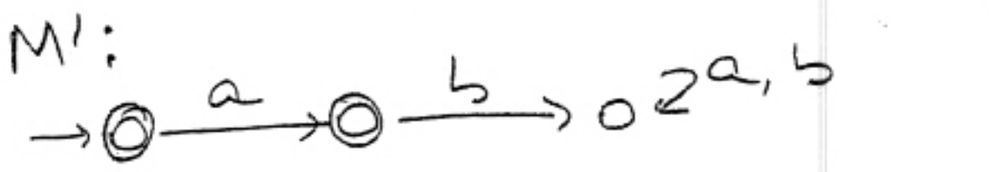
1.14) Mostrar por meio de um exemplo que, se M é um AFN que reconhece a linguagem C, tornando-se estados de aceitação os estados de M que não são de aceitação, e vice-versa, não necessariamente se obtém um novo AFN que reconhece o complemento de C.

Exemplo: AFN para reconhecer palavras sobre o alfabeto {a, b} que começam com ab. Logo, o complemento desta linguagem é o conjunto de palavras que <u>não</u> começam com ab.

AFN original, M:



AFN com estados de aceitação invertidos, M':



Note que qualquer palavra que começa com b pertence ao complemento da linguagem original, e portanto deveria ser aceito pelo automato modificado M'. Mas note que isso não ocorre, e portanto M' não reconhece o complemento da linguagem original.

Na 2ª parte da questão devemos responder se a classe das linguagens reconhecidas por AFN's é fechada sob complemento. A resposta é <u>afirmativa</u>, pois se uma linguagem é reconhecida por um AFN, então também é reconhecida por um AFD, M_1. Logo, pode-se

P86 ⇒

Formalmente:

$M' = (Q', \Sigma, \delta', q_0, F')$ onde

$Q' = Q \cup \{q_0\}$

q_0 é o estado inicial

$F' = \{q_1\}$

δ' é definida a partir de δ:

$$\delta(p, a) = q \Rightarrow \delta'(q, a) = p$$

$$\delta'(q_0, \varepsilon) = F$$

Note que este autômato reconhece A^R. De fato, se $w \in A$, então existe uma seqüência de estado $r_0 \ldots r_n$ em M como descrito acima. Logo, para $w^R = w_n \ldots w_1$ existirá em M' esta mesma seqüência de estados em ordem reversa, ou seja, $r_n r_{n-1} \ldots r_0$, satisfazendo:

$r_n \in F$ (note que M' pode escolher não deterministicamente r_n, a partir de q_0).

$$r_{i-1} = \delta'(r_i, w_i), \quad i = 1 \ldots n$$

$$r_0 = q_1 \in F'$$

Como r_0 é um estado de aceitação em M', a palavra w^R será também aceita por M'. Portanto, M' reconhece o reverso de A, e nossa prova está concluída.

1.24 Sequencia de estados visitados e saída produzida:

b) T_1 sobre 2 1 1:

estados: q_1 q_2 q_2 q_2

saída: 1 1 1

c) T_1 sobre 1 2 1:

estados: q_1 q_1 q_2 q_2

saída: 0 1 1

f) T_2 sobre b b a b.

estados: q_1 q_3 q_2 q_3 q_2

saída: 1 1 1 1

g) T_2 sobre b b b b b b

estados: q_1 q_3 q_2 q_1 q_3 q_2 q_1

saída: 1 1 0 1 1 0

1.27