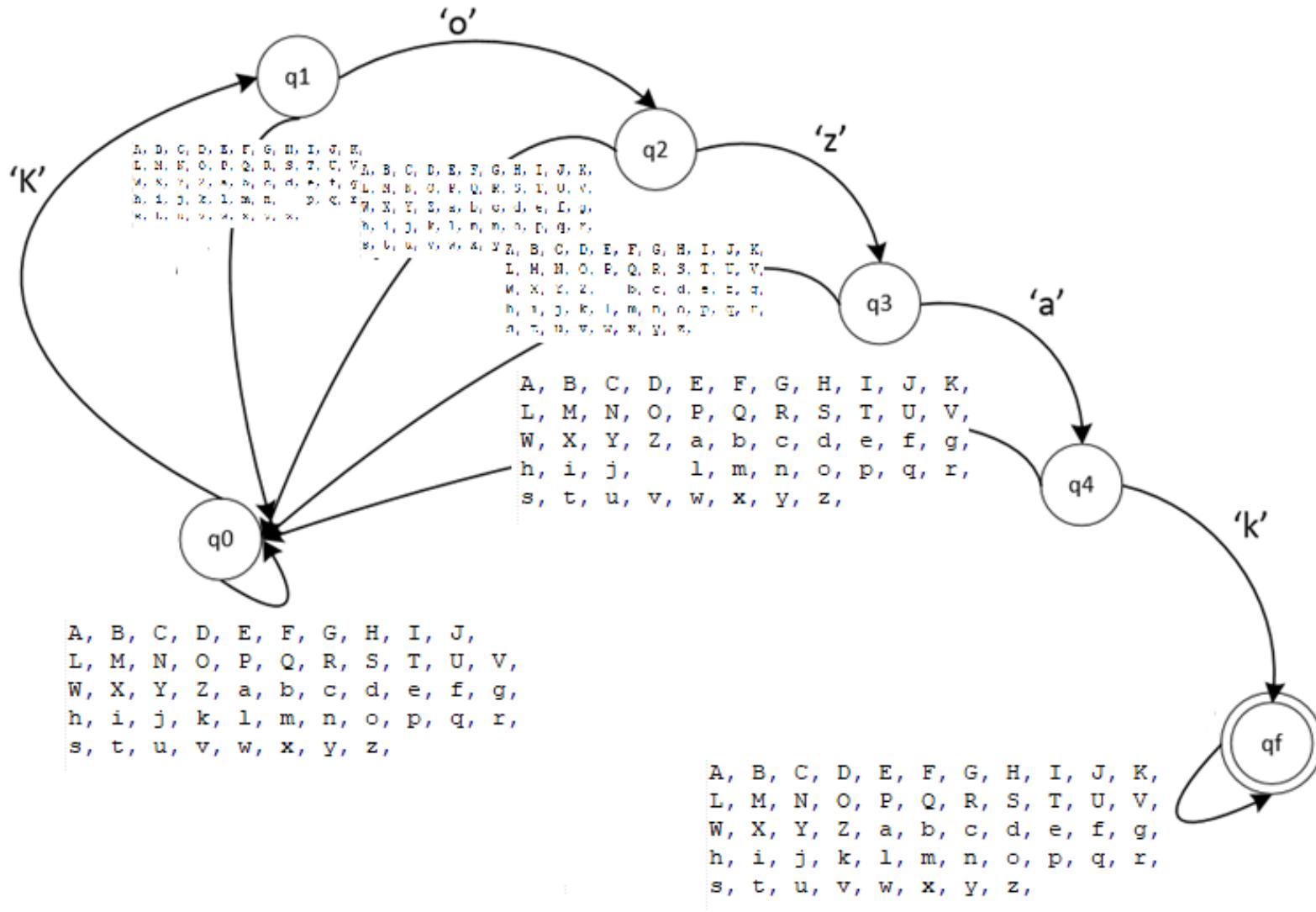


4.3.5. Пошук підрядків за допомогою скінчених автоматів.

Пошук підрядків за допомогою скінчених автоматів (finite automaton pattern matching) - це алгоритм пошуку підрядка в тексті, який використовує скінчений автомат для визначення можливих позицій початку підрядка в тексті.

У цьому алгоритмі вхідний підрядок трансформується в скінчений автомат, що розпізнає його. Далі текст, в якому потрібно знайти підрядок, прочитується посимвольно, і кожен символ використовується для переходу від одного стану автомата до іншого. Якщо після переходу автомат потрапив у фінальний стан, то це означає, що підрядок був знайдений в тексті.

Одним з переваг цього алгоритму є те, що час пошуку не залежить від довжини тексту, а залежить тільки від довжини шуканого підрядка. Однак, побудова скінченого автомата для підрядка може зайняти час і пам'ять, тому використання цього алгоритму може бути ефективним тільки для пошуку одного і того ж підрядка в багатьох текстах.



```
#include "stdio.h"

#define DECLSTATE(NAME, ...) typedef enum {__VA_ARGS__, size##NAME} NAME;
#define GET_ENUM_SIZE(NAME) size##NAME

DECLSTATE(A,
          vA/*A*/,
          vB/*B*/,
          vC/*C*/,
          vD/*D*/,
          vE/*E*/,
          vF/*F*/,
          vG/*G*/,
          vH/*H*/,
          vI/*I*/,
          vJ/*J*/,
          vK/*K*/,
          vL/*L*/,
          vM/*M*/,
          vN/*N*/,
          vO/*O*/,
          vP/*P*/,
          vQ/*Q*/,
          vR/*R*/,
          vS/*S*/,
          vT/*T*/,
          vU/*U*/,
          vV/*V*/,
          vW/*W*/,
          vX/*X*/,
          vY/*Y*/,
          vZ/*Z*/,
          sZ/*[*/,
          sY/*\*/,
          sC/*]*/,
          sH/*^*/,
          sB/*_*/,
          sT/*`*/,
          va/*a*/,
          vb/*b*/,
          vc/*c*/,
          vd/*d*/,
          ve/*e*/,
```

```
vf/*f*/,
vg/*g*/,
vh/*h*/,
vi/*i*/,
vj/*j*/,
vk/*k*/,
vl/*l*/,
vm/*m*/,
vn/*n*/,
vo/*o*/,
vp/*p*/,
vq/*q*/,
vr/*r*/,
vs/*s*/,
vt/*t*/,
vu/*u*/,
vv/*v*/,
vw/*w*/,
vx/*x*/,
vy/*y*/,
vz/*z*/,
sF/*end mark*/
)

DECLSTATE(Q,
q0,
q1,
q2,
q3,
q4,
qf
)
```

```
typedef unsigned char INSTRUCTION;
typedef INSTRUCTION PROGRAM[GET_ENUM_SIZE(A)][GET_ENUM_SIZE(Q)];

PROGRAM program = {
    //   q0  q1  q2  q3  q4  qf
    /*A*/{q0, q0, q0, q0, q0, qf},
    /*B*/{q0, q0, q0, q0, q0, qf},
    /*C*/{q0, q0, q0, q0, q0, qf},
    /*D*/{q0, q0, q0, q0, q0, qf},
    /*E*/{q0, q0, q0, q0, q0, qf},
    /*F*/{q0, q0, q0, q0, q0, qf},
    /*G*/{q0, q0, q0, q0, q0, qf},
    /*H*/{q0, q0, q0, q0, q0, qf},
    /*I*/{q0, q0, q0, q0, q0, qf},
    /*J*/{q0, q0, q0, q0, q0, qf},
    /*K*/{q1, q0, q0, q0, q0, qf},
    /*L*/{q0, q0, q0, q0, q0, qf},
    /*M*/{q0, q0, q0, q0, q0, qf},
    /*N*/{q0, q0, q0, q0, q0, qf},
    /*O*/{q0, q0, q0, q0, q0, qf},
    /*P*/{q0, q0, q0, q0, q0, qf},
    /*Q*/{q0, q0, q0, q0, q0, qf},
    /*R*/{q0, q0, q0, q0, q0, qf},
    /*S*/{q0, q0, q0, q0, q0, qf},
    /*T*/{q0, q0, q0, q0, q0, qf},
    /*U*/{q0, q0, q0, q0, q0, qf},
    /*V*/{q0, q0, q0, q0, q0, qf},
    /*W*/{q0, q0, q0, q0, q0, qf},
    /*X*/{q0, q0, q0, q0, q0, qf},
    /*Y*/{q0, q0, q0, q0, q0, qf},
    /*Z*/{q0, q0, q0, q0, q0, qf},
    /*[*/{q0, q0, q0, q0, q0, qf},
```

```
/*\*/{q0, q0, q0, q0, q0, qf},  
/*]/*/{q0, q0, q0, q0, q0, qf},  
/*^*/{q0, q0, q0, q0, q0, qf},  
/*_*/{q0, q0, q0, q0, q0, qf},  
/*`*/{q0, q0, q0, q0, q0, qf},  
/*a*/{q0, q0, q0, q4, q0, qf},  
/*b*/{q0, q0, q0, q0, q0, qf},  
/*c*/{q0, q0, q0, q0, q0, qf},  
/*d*/{q0, q0, q0, q0, q0, qf},  
/*e*/{q0, q0, q0, q0, q0, qf},  
/*f*/{q0, q0, q0, q0, q0, qf},  
/*g*/{q0, q0, q0, q0, q0, qf},  
/*h*/{q0, q0, q0, q0, q0, qf},  
/*i*/{q0, q0, q0, q0, q0, qf},  
/*j*/{q0, q0, q0, q0, q0, qf},  
/*k*/{q0, q0, q0, q0, qf, qf},  
/*l*/{q0, q0, q0, q0, q0, qf},  
/*m*/{q0, q0, q0, q0, q0, qf},  
/*n*/{q0, q0, q0, q0, q0, qf},  
/*o*/{q0, q2, q0, q0, q0, qf},  
/*p*/{q0, q0, q0, q0, q0, qf},  
/*q*/{q0, q0, q0, q0, q0, qf},  
/*r*/{q0, q0, q0, q0, q0, qf},  
/*s*/{q0, q0, q0, q0, q0, qf},  
/*t*/{q0, q0, q0, q0, q0, qf},  
/*u*/{q0, q0, q0, q0, q0, qf},  
/*v*/{q0, q0, q0, q0, q0, qf},  
/*w*/{q0, q0, q0, q0, q0, qf},  
/*x*/{q0, q0, q0, q0, q0, qf},  
/*y*/{q0, q0, q0, q0, q0, qf},  
/*z*/{q0, q0, q3, q0, q0, qf}  
};
```

```
typedef struct structDFA{
    unsigned char * data;
    PROGRAM * program;
    void(*run)(struct structDFA * dfa);
    Q state;
} DFA;

void runner(DFA * dfa){
    for (; *dfa->data != sF; ++dfa->data){
        dfa->state = (*dfa->program)[*dfa->data][dfa->state];
    }
}

#define MAX_TEXT_SIZE 256
int main(){
    unsigned char data[MAX_TEXT_SIZE] = "Kozak Nazar Bohdanovych", *data_ = data;
    DFA dfa = { data, (PROGRAM*)program, runner, q0 };

    for (; *data_; *data_ -= 'A', *data_ %= GET_ENUM_SIZE(A), ++data_);
    *data_ = sF;
    dfa.run(&dfa);

    if (dfa.state == qf){
        printf("DFA: finit state\r\n");
    }
    else{
        printf("DFA: no finit state\r\n");
    }

    getchar();
    return 0;
}
```

```
#include "stdio.h"

#define DECLSTATE(NAME, ...) typedef enum {__VA_ARGS__, size##NAME} NAME;
#define GET_ENUM_SIZE(NAME) size##NAME

DECLSTATE(A,
          vA/*A*/,
          vB/*B*/,
          vC/*C*/,
          vD/*D*/,
          vE/*E*/,
          vF/*F*/,
          vG/*G*/,
          vH/*H*/,
          vI/*I*/,
          vJ/*J*/,
          vK/*K*/,
          vL/*L*/,
          vM/*M*/,
          vN/*N*/,
          vO/*O*/,
          vP/*P*/,
          vQ/*Q*/,
          vR/*R*/,
          vS/*S*/,
          vT/*T*/,
          vU/*U*/,
          vV/*V*/,
          vW/*W*/,
          vX/*X*/,
          vY/*Y*/,
          vZ/*Z*/,
          sZ/*[*|,
          sY/*\*/|,
          sC/*]*/|,
          sH/*^*/|,
          sB/*_*|,
          sT/*`*/|,
          va/*a*/,
          vb/*b*/,
          vc/*c*/,
          vd/*d*/,
          ve/*e*/,
          vf/*f*/.
```

```
vg/*g*/,
vh/*h*/,
vi/*i*/,
vj/*j*/,
vk/*k*/,
vl/*l*/,
vm/*m*/,
vn/*n*/,
vo/*o*/,
vp/*p*/,
vq/*q*/,
vr/*r*/,
vs/*s*/,
vt/*t*/,
vu/*u*/,
vv/*v*/,
vw/*w*/,
```

```

vx/*x*/,
vy/*y*/,
vz/*z*/,
sF/*end mark*/
)

DECLSTATE(Q,
q0,
q1,
q2,
q3,
q4,
qf
)

typedef unsigned char INSTRUCTION;
#define MAX_RULE_COUNT 16
typedef struct {
    unsigned char input;
    INSTRUCTION toState;
} SimpleRule;
typedef SimpleRule PROGRAM[GET_ENUM_SIZE(Q)][MAX_RULE_COUNT];

#define DEFAULT GET_ENUM_SIZE(A)

PROGRAM program = {
    /           q0           q1           q2
q3           q4           qf
    { { vK, q1 }, { DEFAULT, q0 } }, { { vo, q2 }, { DEFAULT, q0 } }, { { vz, q3 }, { DEFAULT, q0 } },
    { { va, q4 }, { DEFAULT, q0 } }, { { vk, qf }, { DEFAULT, q0 } }, { { DEFAULT, qf } },
};

typedef struct structDFA{
    unsigned char * data;
    PROGRAM * program;
    void(*run)(struct structDFA * dfa);
    Q state;
} DFA;

```

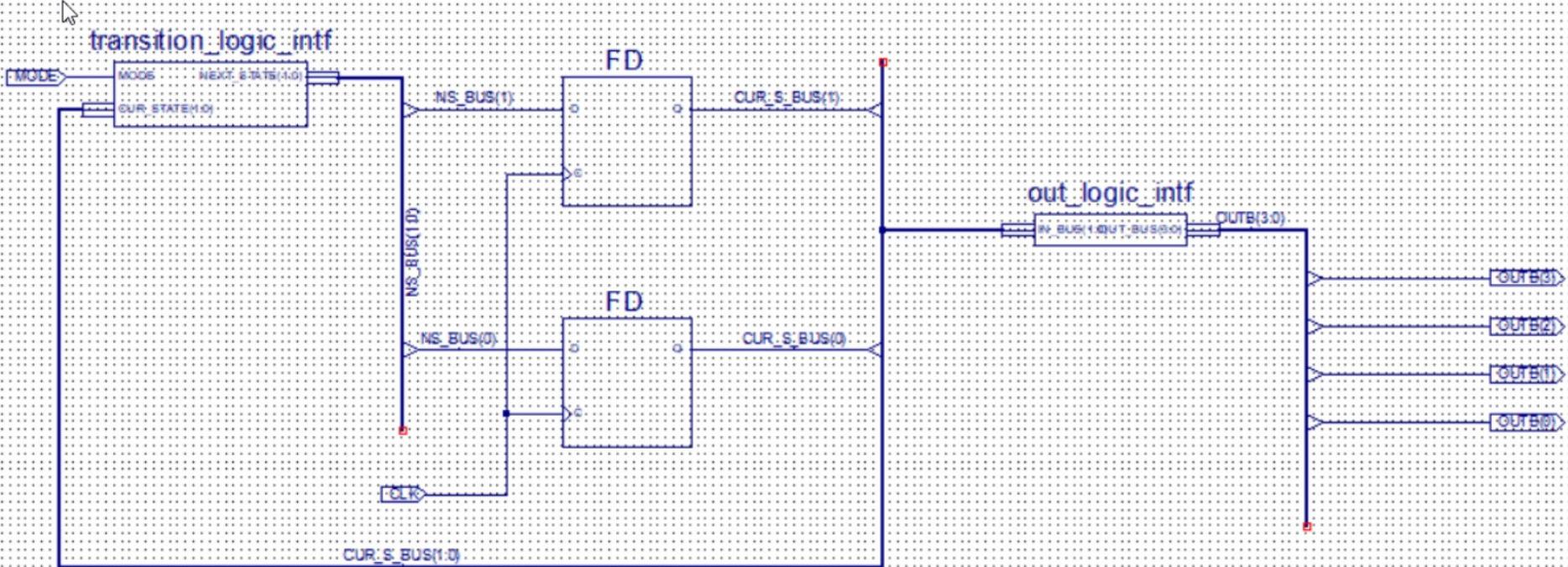
```
void runner(DFA * dfa){
    for (; *dfa->data != sF; ++dfa->data){
        SimpleRule *prevRule, *rule;
        for (prevRule = rule = (*dfa->program)[dfa->state]; prevRule->input != DEFAULT; prevRule = rule, ++rule){
            if (DEFAULT == rule->input){
                dfa->state = rule->toState;
                break;
            }
            else if (*dfa->data == rule->input){
                dfa->state = rule->toState;
                break;
            }
        }
    }
}

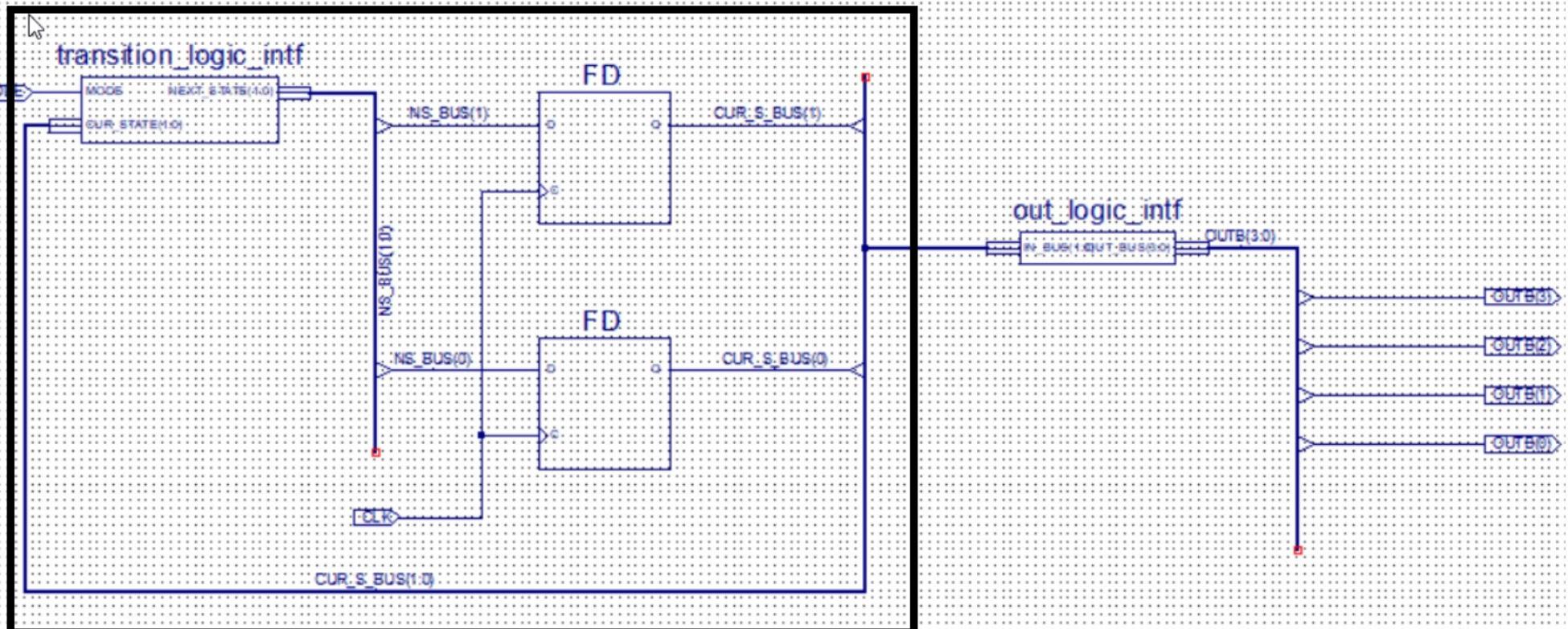
#define MAX_TEXT_SIZE 256
int main(){
    unsigned char data[MAX_TEXT_SIZE] = "Kozak Nazar Bohdanovych", *data_ = data;
    DFA dfa = { data, (PROGRAM*)program, runner, q0 };

    for (; *data_; *data_ -= 'A', *data_ %= GET_ENUM_SIZE(A), ++data_);
        *data_ = sF;
    dfa.run(&dfa);

    if (dfa.state == qf){
        printf("DFA: finit state\r\n");
    }
    else{
        printf("DFA: no finit state\r\n");
    }

    getchar();
    return 0;
}
```

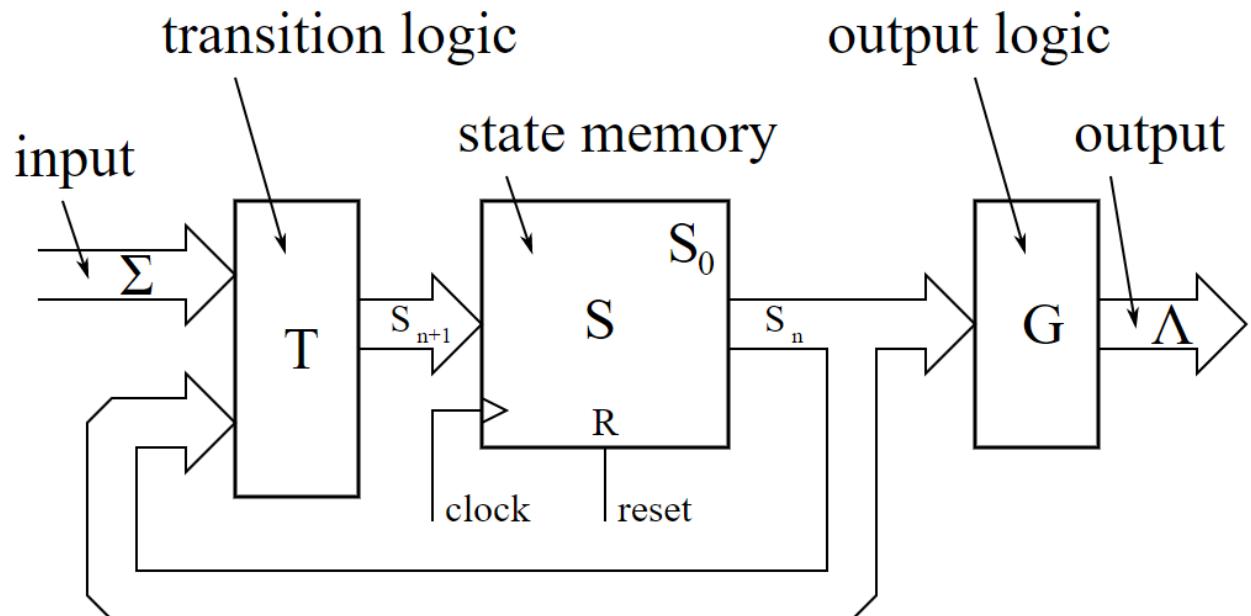


1.2.5. Перетворювачі(трансдуктори) на основі детермінованого скінченного автомата.

1.2.5.1. Автомат Мура(Moore machine).

Автомат Мура це 6 елементів (S , S_0 , Σ , Λ , T , G):

- множина внутрішніх станів S (внутрішній алфавіт);
- початковий стан S_0 , який є елементом (S);
- скінчена множина вхідних сигналів Σ (вхідний алфавіт);
- скінчена множина вихідних сигналів Λ (вихідний алфавіт);
- функція переходу ($T: S \times \Sigma \rightarrow S$), яка відображає стан і вхідний алфавіт у наступний стан;
- виходна функція ($G: S \rightarrow \Lambda$), яка відображає стан у вихідний алфавіт.

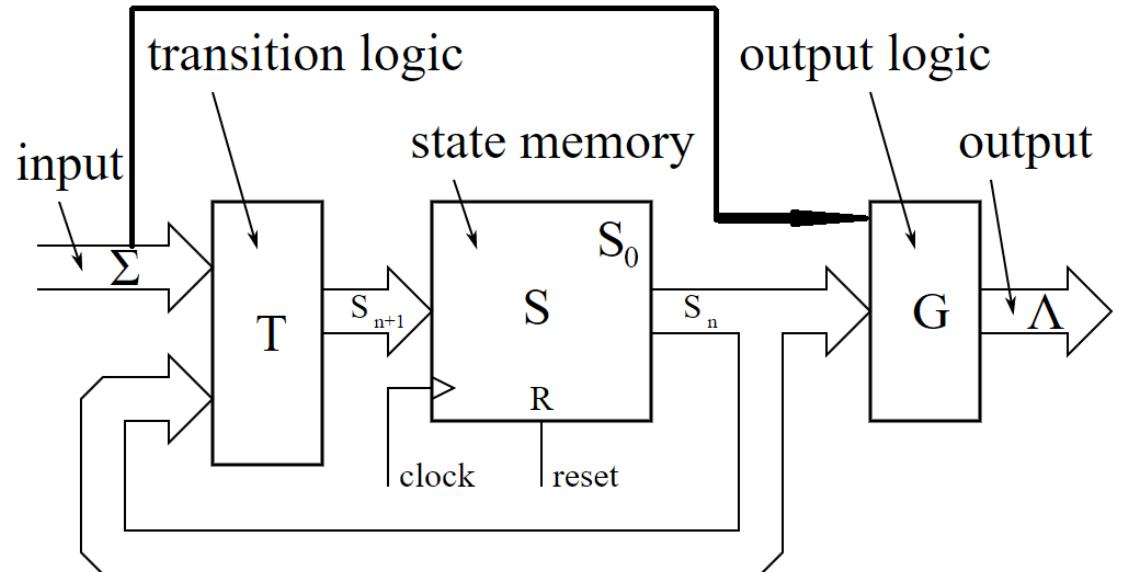


1.2.5. Перетворювачі(трансдуктори) на основі детермінованого скінченного автомата.

1.2.5.2. АВТОМАТ МІЛІ(Mealy machine).

Аutomat Mіlі це 6 елементів (S , S_0 , Σ , Λ , T , G):

- множина внутрішніх станів S (внутрішній алфавіт);
- початковий стан S_0 , який є елементом (S);
- скінчена множина вхідних сигналів Σ (вхідний алфавіт);
- скінчена множина вихідних сигналів Λ (вихідний алфавіт);
- функція переходу ($T: S \times \Sigma \rightarrow S$), яка відображає стан і вхідний алфавіт у наступний стан;
- вихідна функція ($G: S \rightarrow \Lambda$), яка відображає стан і вхідний алфавіт у вихідний алфавіт.



Говорять, що скінчений автомат $M = (S, I, f, s_0, F)$ **допускає** (приймає) ланцюжок α , якщо він переводить початковий стан s_0 в заключний стан; це означає, що стан $f(s_0, \alpha)$ – елемент множини F .

Мова $L(M)$, яку **розвізнає автомат** M , – це множина всіх ланцюжків, які допускає автомат M . Два автомати називають **еквівалентними**, якщо вони розвінюють одну й ту саму мову.

Якщо визначити **слово** як стрічку символів, що створена через конкатенацію (з'єднання), а **мову** як множину слів (може бути нескінченною множиною), то говорять, що мова L читається (приймається) автомatem M , якщо по закінченню обробки він переходить в один з кінцевих станів F .

Ієрархія Чомскі, або Ієрархія Чомскі-Шутценберг'ера (названа на честь мовознавця Ноама Чомскі та математика Марселя Шутценберг'ера) — ієрархія формальних граматик, які породжують формальні мови. Вперше описана Ноамом Чомскі в 1956 році. Чотири описані Чомскі типи граматик виходять від базової, необмеженої граматики (граматика типу 0), на яку послідовно накладають обмеження на правила продукції.

В залежності від типу найпростішої граматики, яка може згенерувати задану формальну мову, формальні мови ділять на відповідні категорії від типу 0 до типу 3.

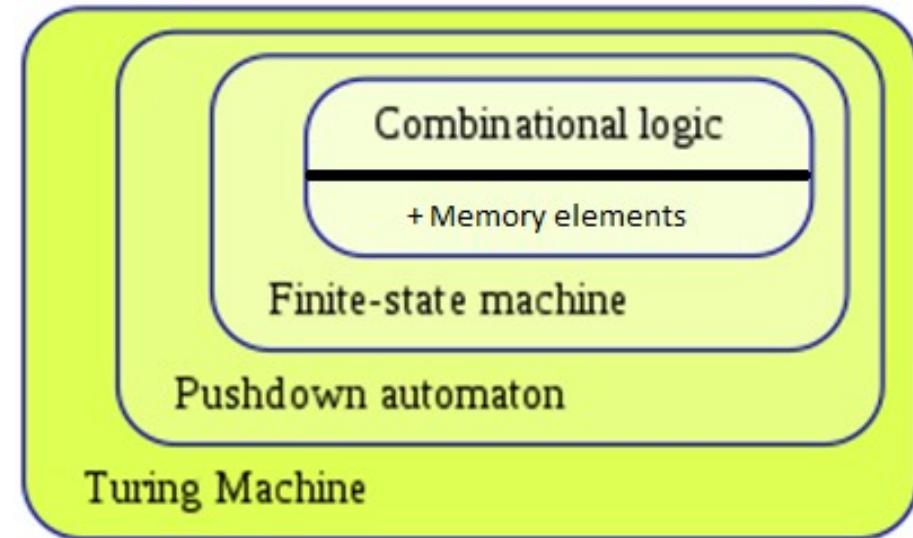
Граматика	Правила	Мови	Автомати	Скорочення
Тип-0 <u>Довільна</u> <u>формальна</u> граматика	$\alpha \rightarrow \beta$ $\alpha \in V^* N V^*, \beta \in V^*$	<u>рекурсивно</u> <u>зліченна</u>	<u>Машина Тюринга</u>	KSV*
Тип-1 <u>Контекстно-</u> <u>залежна</u> граматика	$\alpha A \beta \rightarrow \alpha'y'\beta$ $A \in N, \alpha, \beta \in V^*, y \in V^+$ S $\rightarrow \epsilon$ дозволене, коли серед правил P відсутнє $\alpha \rightarrow \beta S y$.	<u>контекстно-</u> <u>залежна</u>	<u>Лінійний обмежений автомат</u>	КЗ
Тип-2 <u>Контекстно-</u> <u>вільна</u> граматика	$A \rightarrow y$ $A \in N, y \in V^*$	<u>контекстно-</u> <u>вільна</u>	недетермінований автомат з магазинною пам'ятю	КВ
Тип-3 <u>регулярна</u> граматика	$S \rightarrow \epsilon$ $A \rightarrow aB$ (праволінійна) або $A \rightarrow Ba$ (ліволінійна) $A \rightarrow a$ $A \rightarrow \epsilon$ $A, B \in N, a \in \Sigma$	<u>регулярна</u>	Скінчений автомат (як детермінований, так і недетермінований)	А

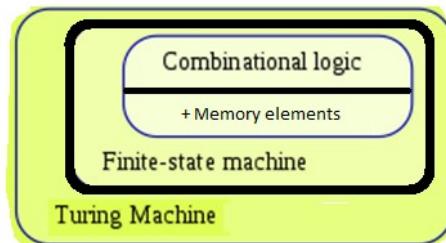
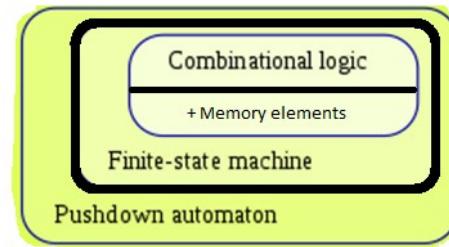
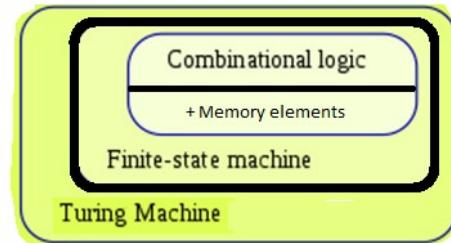
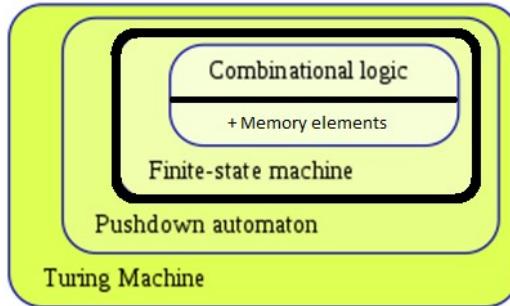
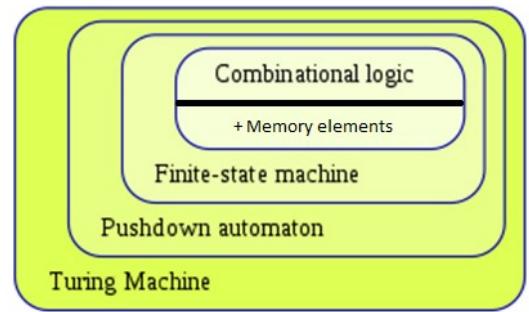
recursively enumerable

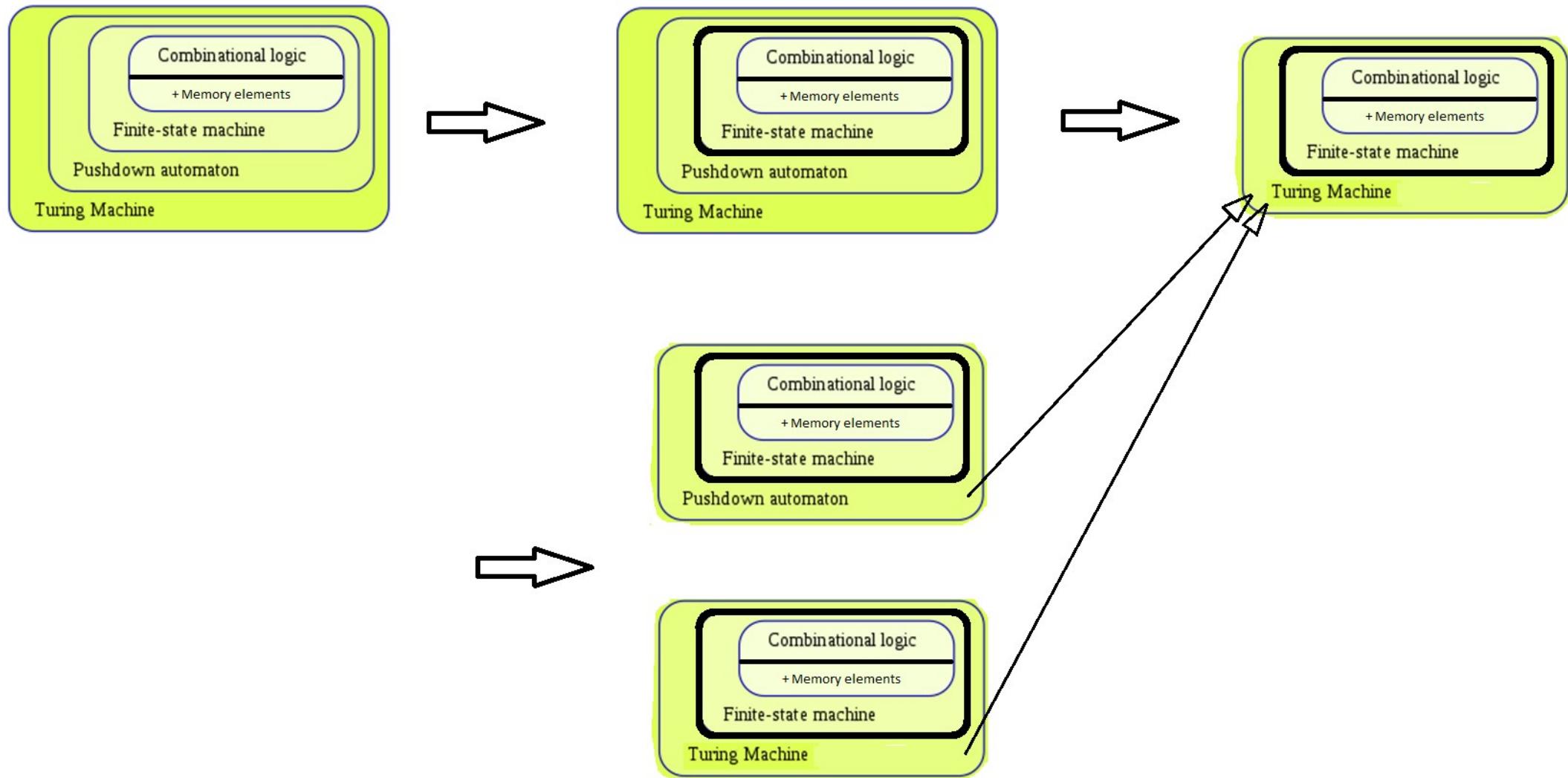
context-sensitive

context-free

regular







Формування автомату з однобуквеними переходами, одним початковим та одним кінцевим станами з довільного скінченного автомата.

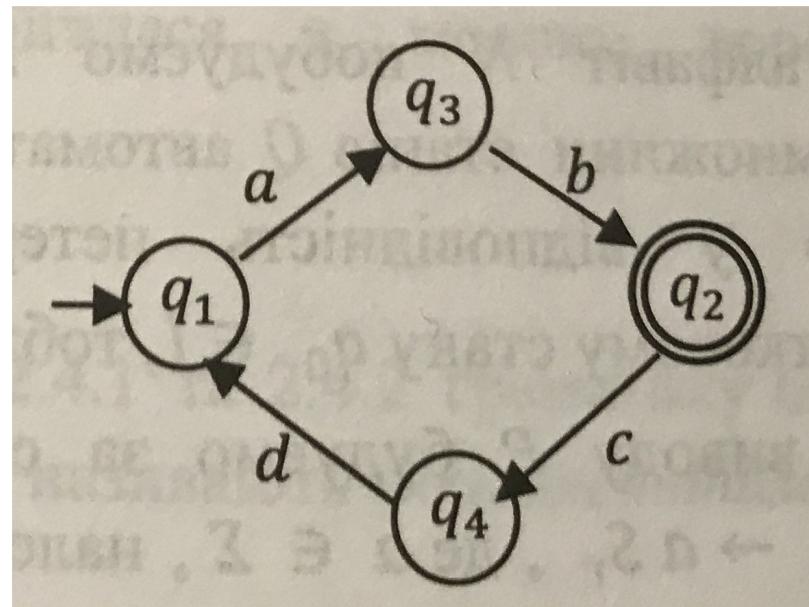
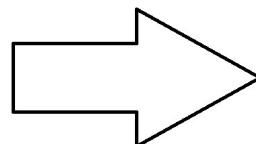
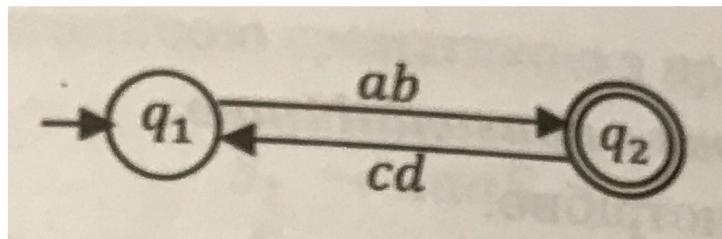
1. За наявності декількох початкових станів автомата додати додатковий стан, який стане єдиним початковим станом, та з'єднати його дугами з міткою λ з кожним з попередніх початкових станів, при цьому попередні початкові стани надалі вважаються звичайними станами.
2. За наявності декількох заключних станів автомата, необхідно додати новий стан, який стане надалі єдиним заключним станом, та з'єднати кожен попередній заключний стан з новим дугою з міткою λ . Попередні заключні стани стають при цьому звичайними станами.
3. За наявності багатобуквеної мітки $a_1 \dots a_i$ на дузі між двома станами p та q , необхідно додати додаткові стани p_1, \dots, p_{i-1} , які будуть проміжними між станами p та q , та з'єднати стан p з станом p_1 дугою з міткою a_1 , стан p_1 з станом p_2 дугою з міткою a_2 і т. д.

Формування автомату з однобуквеними переходами з довільного скінченного автомата.

Приклад:

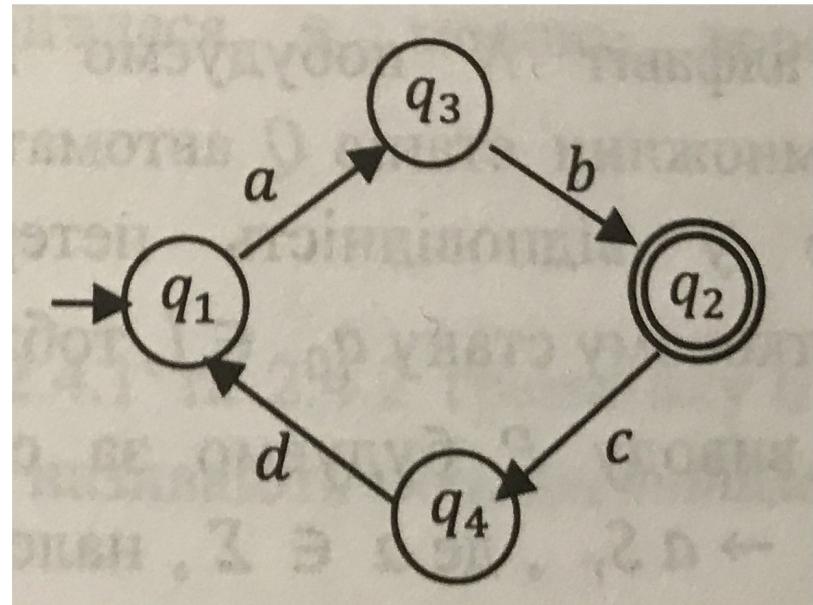
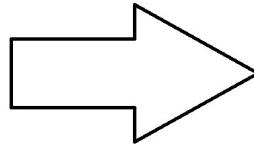
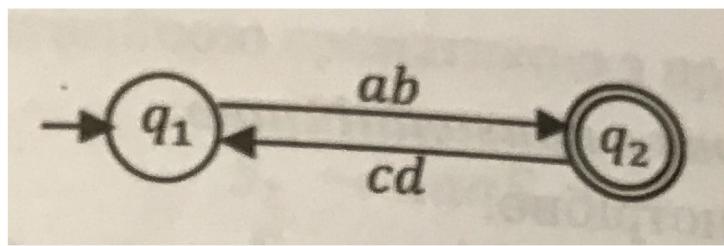
$M = (Q, \Sigma, \Delta, I, F)$, де

$Q = \{q_1, q_2\}$, $\Sigma = \{a, b, c, d\}$, $I = \{q_1\}$, $F = \{q_2\}$, $\Delta = \{ \langle q_1, ab, q_2 \rangle, \langle q_2, cd, q_1 \rangle \}$.



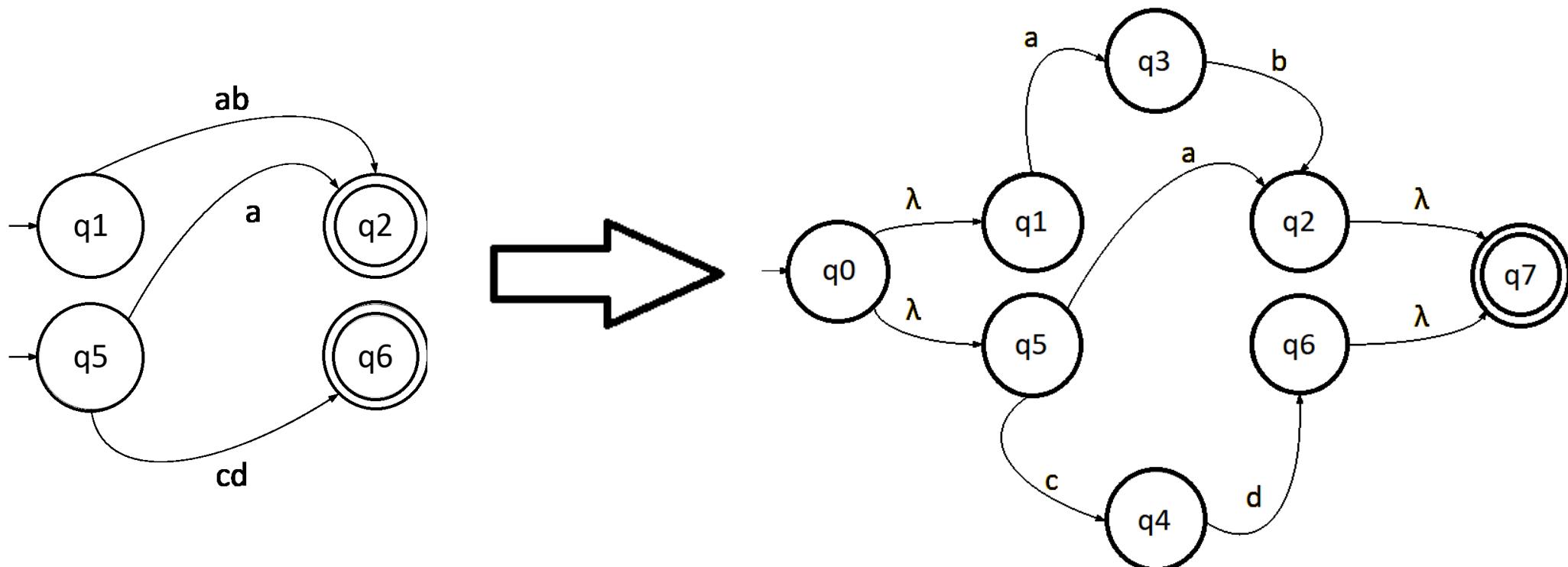
Формування автомату з однобуквеними переходами з довільного скінченного автомата.

Приклад:



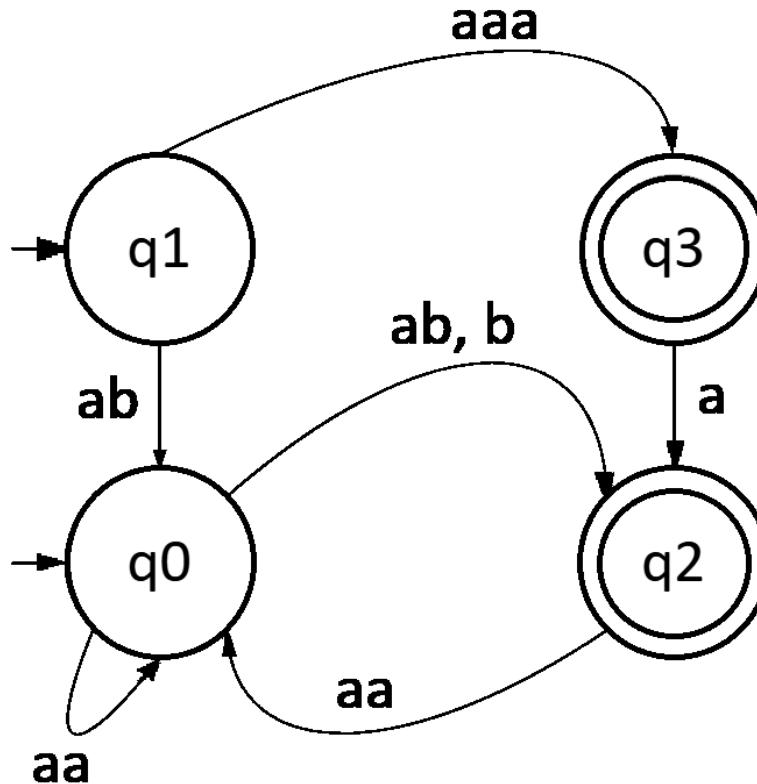
Формування автомату з однобуквеними переходами, одним початковим та одним кінцевим станами з довільного скінченного автомата.

Приклад 2:



Контрольне завдання №9

Сформувати автомат з однобуквеними переходами, одним початковим та одним кінцевим станами з заданого недетермінованого скінченного автомата.



$$G = (N, T, P, S)$$

Мовою $L(G)$, що породжується граматикою G , називають множину всіх ланцюжків термінальних символів, які виводяться з початкового символу S , тобто

$$L(G) = \left\{ \omega \in T^* \mid S \xrightarrow{*} \omega \right\}.$$

Знаходження мови, що породжується заданою граматикою. Приклад:

$$G = (N = \{S, A\},$$

$$T = \{a, b\},$$

$$P = \{S \rightarrow aA,$$

$$S \rightarrow b,$$

$$A \rightarrow aa\},$$

аксіома граматики S)

Для цього запишемо вивід всіх можливих ланцюжків з початкового символу S , послідовно заміняючи кожен нетермінал за допомогою існуючих продукцій, поки не отримаємо ланцюжок з терміналів:

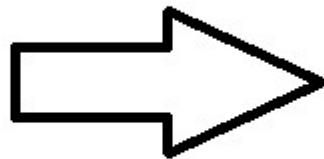
$$S \Rightarrow aA \Rightarrow aaa;$$

$$S \Rightarrow b.$$

$$\text{Отже } L(G) = \{aaa, b\}.$$

Знаходження мови, що породжується заданою граматикою. Приклад:

$$G = (N = \{S, A\},$$
$$T = \{a, b\},$$
$$P = \{ S \rightarrow aA,$$
$$S \rightarrow b,$$
$$A \rightarrow aa\},$$
$$S)$$



$$S \Rightarrow aA \Rightarrow aaa;$$
$$S \Rightarrow b.$$
$$L(G) = \{aaa, b\} .$$

Знаходження скінченного автомата еквівалентного граматиці.

$G = (N, T, P, S)$, $N = \{A, B, S\}$, $T = \{a, b\}$, S – початковий символ,

$P = \{S \rightarrow aA,$

$A \rightarrow aA,$

$A \rightarrow a,$

$S \rightarrow \varepsilon,$

$S \rightarrow bB,$

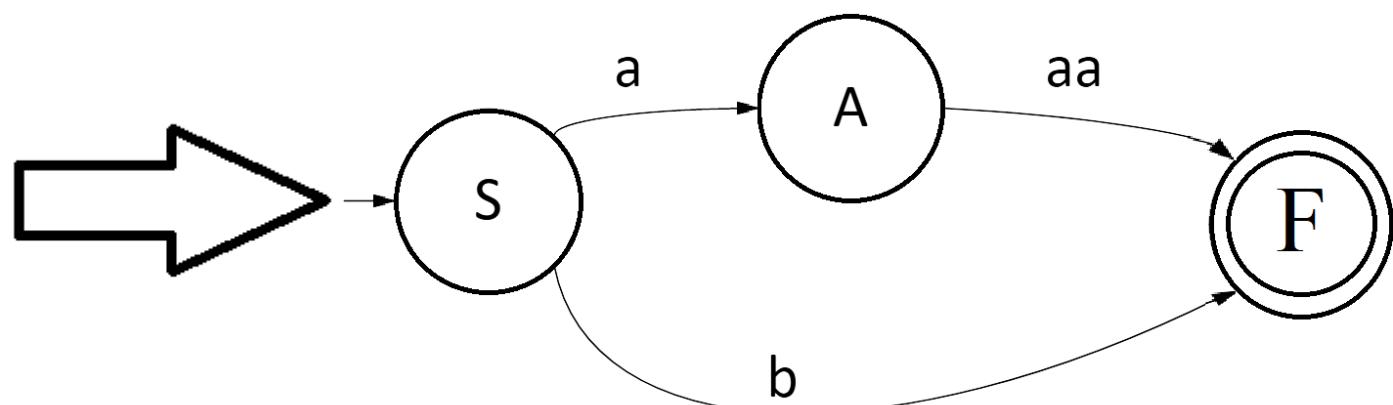
$B \rightarrow a\}$.

Побудуємо скінчений автомат, еквівалентний граматиці. Множину термінальних символів зробимо вхідним алфавітом автомата $\Sigma = \{a, b\}$. Множині нетермінальних символів $N = \{A, B, S\}$ поставимо у відповідність множину станів $Q = \{q_1, q_2, q_0, q_3\}$, для чого правила вигляду $A \rightarrow a$ замінимо на два правила: $A \rightarrow aF$, $F \rightarrow \varepsilon$, де F – новий додатково введений нетермінал, $N = \{A, B, S, F\}$. Початковий символ S зробимо відповідним початковому стану q_0 , зробимо заключним стан q_3 , він відповідає нетерміналові F , для якого задано ε – правило $F \rightarrow \varepsilon$. Продукціям граматики поставимо у відповідність переходи автомата, враховуючи, що q_1 відповідає нетерміналові A , q_2 – нетерміналові B .

$$\Delta = \{(q_0, a, q_1), (q_1, a, q_1), (q_1, a, q_3), (q_0, \varepsilon, q_3), (q_0, b, q_2), (q_2, a, q_3)\}.$$

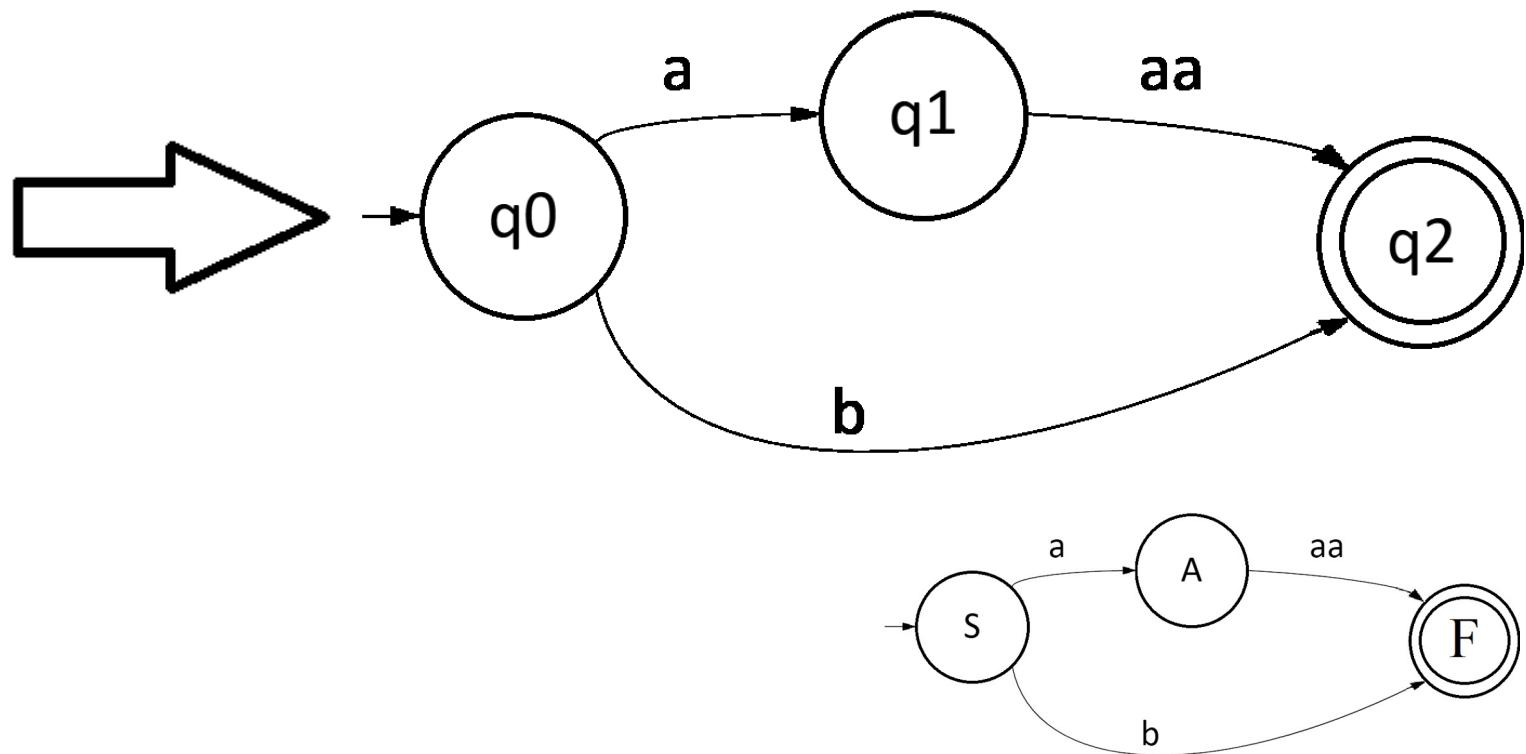
Знаходження скінченного автомата еквівалентного граматиці. Приклад:

$G = (N = \{S, A\},$
 $T = \{a, b\},$
 $P = \{ S \rightarrow aA,$
 $S \rightarrow b,$
 $A \rightarrow aa\},$
 $S)$



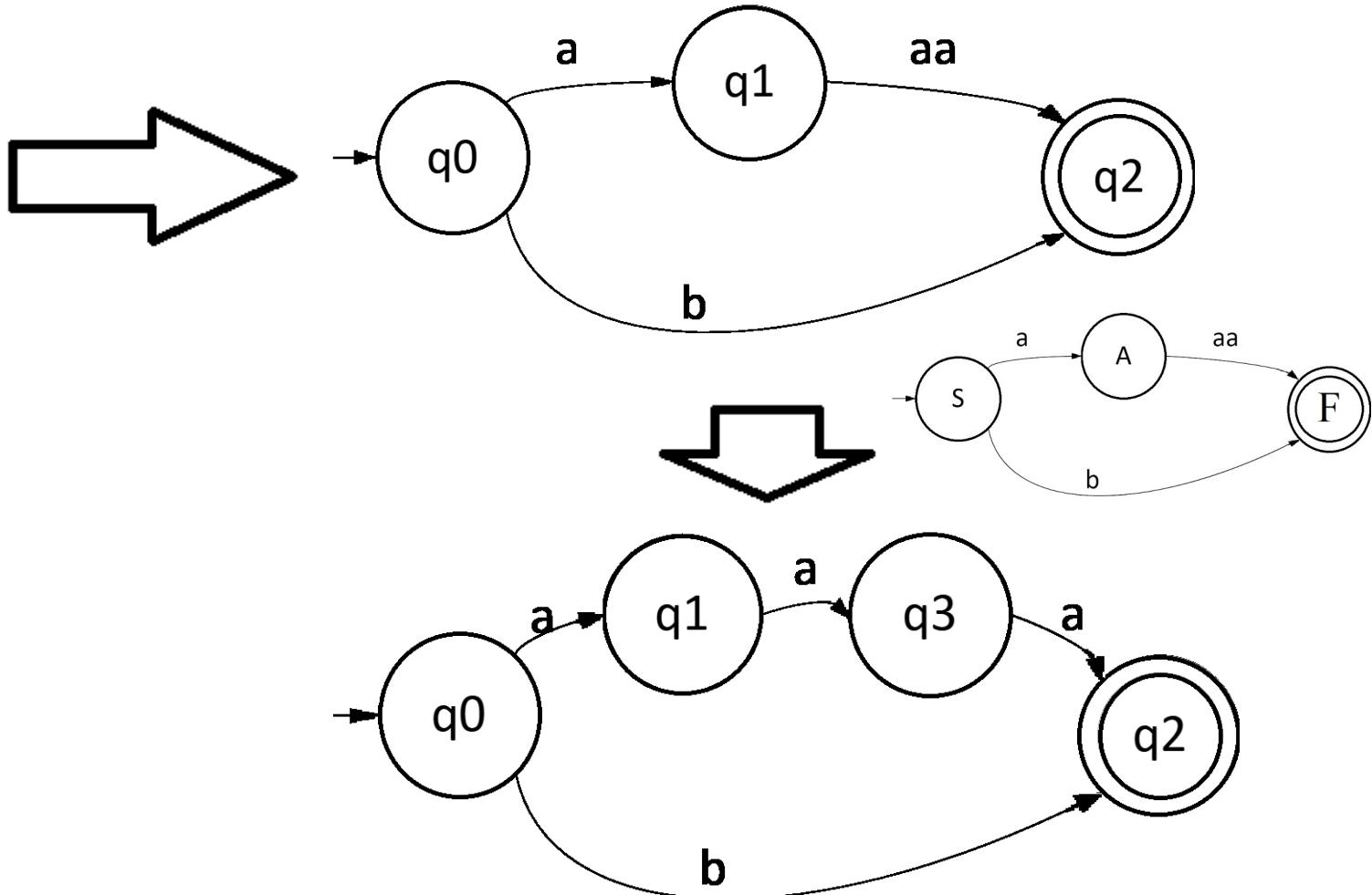
Знаходження скінченного автомата еквівалентного граматиці. Приклад:

$G = (N = \{S, A\},$
 $T = \{a, b\},$
 $P = \{ S \rightarrow aA,$
 $S \rightarrow b,$
 $A \rightarrow aa\},$
 $S)$



Знаходження скінченного автомата еквівалентного граматиці та формування на його основі автомatu з однобуквеними переходами. **Приклад:**

$$G = (N = \{S, A\}, T = \{a, b\}, P = \{S \rightarrow aA, S \rightarrow b, A \rightarrow aa\}, S)$$



Непродуктивні та недосяжні стани скінченного автомата.

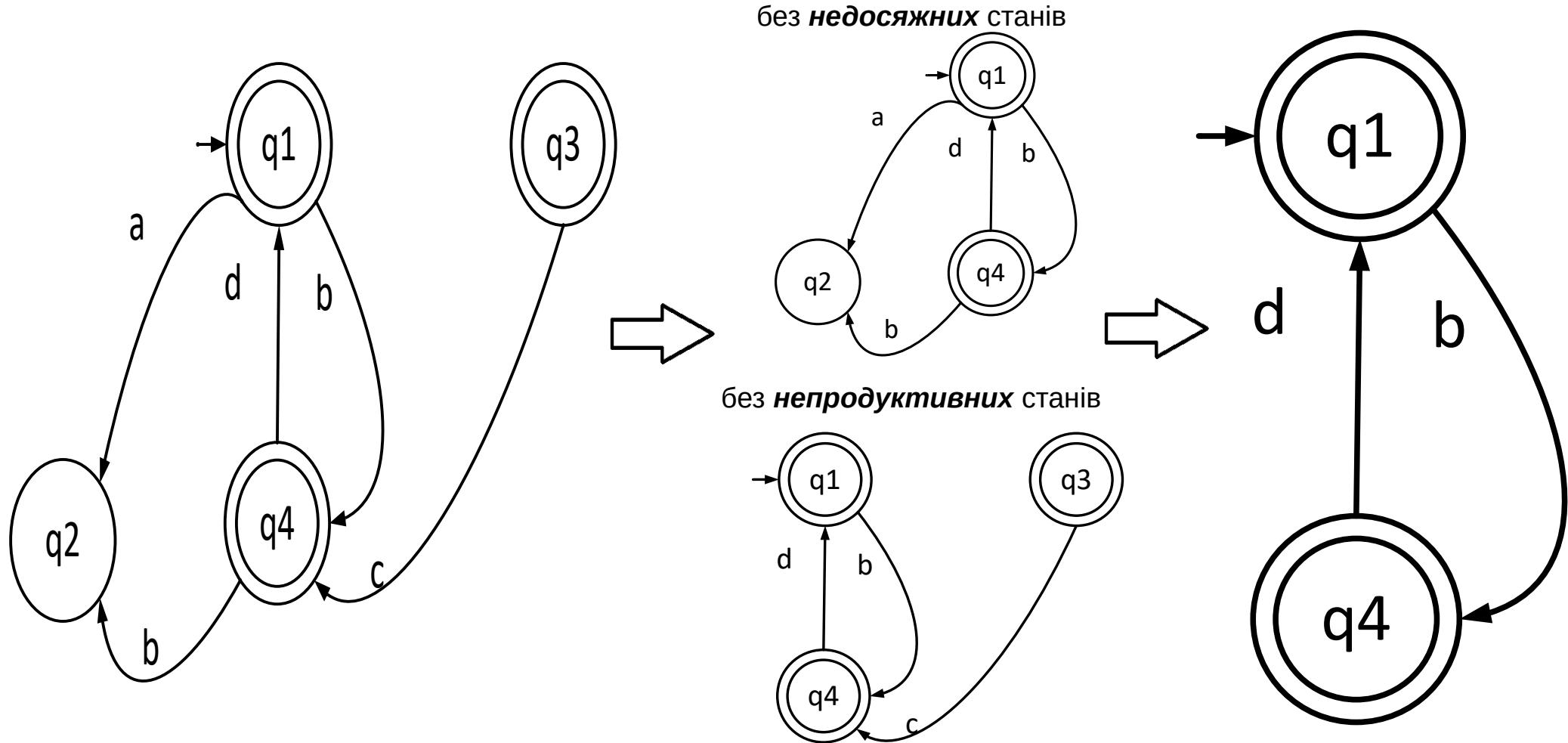
Стан автомата називається **недосяжним**, якщо не існує шляху до нього з початкового стану.

Стан автомата називається **непродуктивним**, якщо не існує шляху від нього до заключного(кінцевого) стану.

Важливе уточнення

**Для перетворювачів (трансдукторів) мінімізацію кінцевих
станів не виконуємо.**

Видалення непродуктивних та недосяжних станів скінченного автомата. **Приклад:**



Видалення λ -переходів

Щоб перейти від вихідного скінченного автомата $M = \langle Q, \Sigma, \Delta, I, F \rangle$ до еквівалентного скінченного автомата $M' = \langle Q', \Sigma, \Delta', I', F' \rangle$ без λ -переходів, достатньо у вихідному графі M здійснити такі перетворення.

1. Всі стани, крім початкового, в які заходять тільки дуги з міткою λ , видаляються; тим самим визначається множина Q' скінченного автомата M' . Зрозуміло, що $Q' \subseteq Q$. При цьому вважаємо, що початковий стан залишається попереднім.

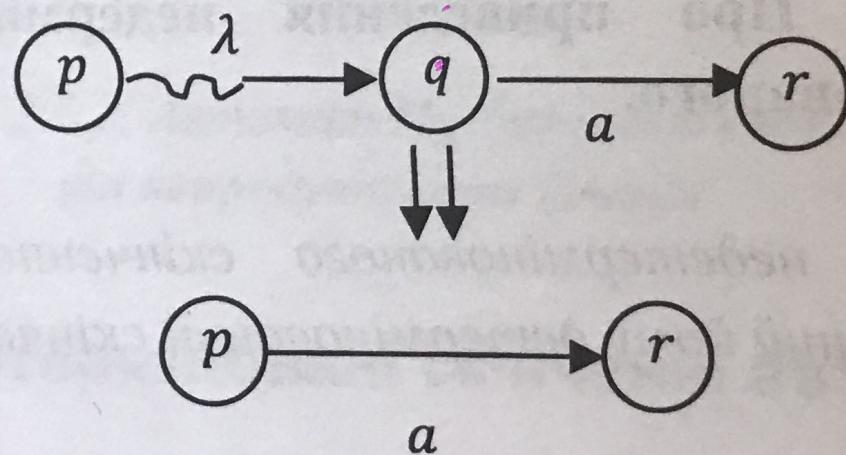
2. Множина дуг скінченного автомата M' та їх міток (тим самим і функція переходів M') визначається так:
 для довільних двох станів $p, r \in Q'$ перехід 3
 p в r по дузі з міткою a :

$$p \xrightarrow{a} r$$

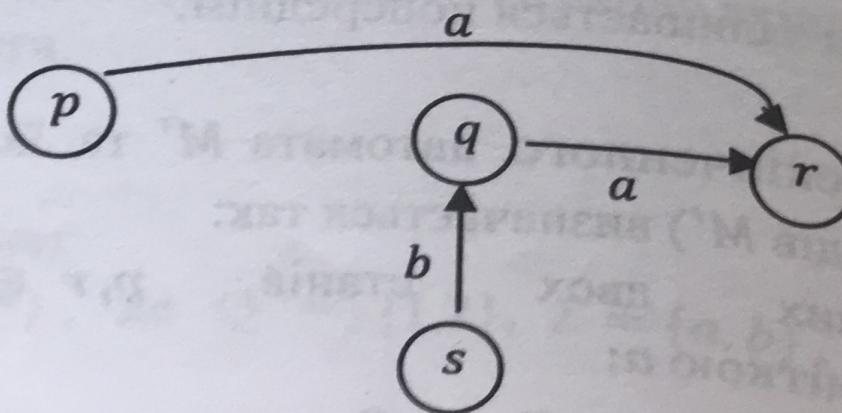
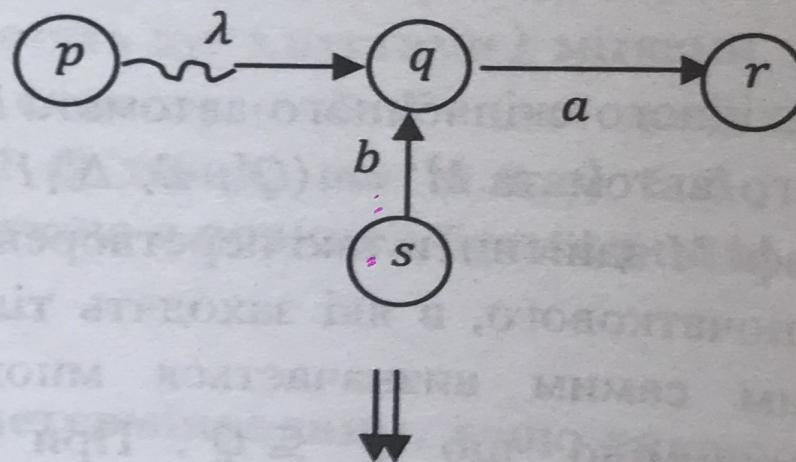
має місце тоді і тільки тоді, коли $a \in \Sigma$, а в графі M
 існує дуга з p в r , мітка якої символ a

або існує такий стан q , що $p \xrightarrow{\lambda}^+ q$ і $q \xrightarrow{a} r$.

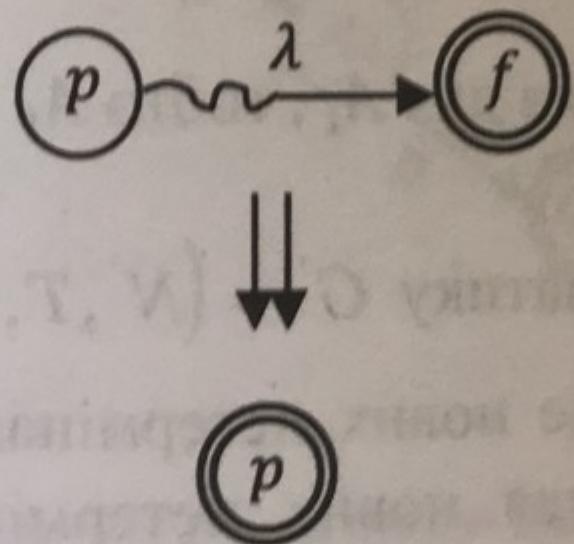
При цьому вершина q , взагалі кажучи, може не належати Q' і надалі бути видалена в процесі подальшого видалення λ переходів.



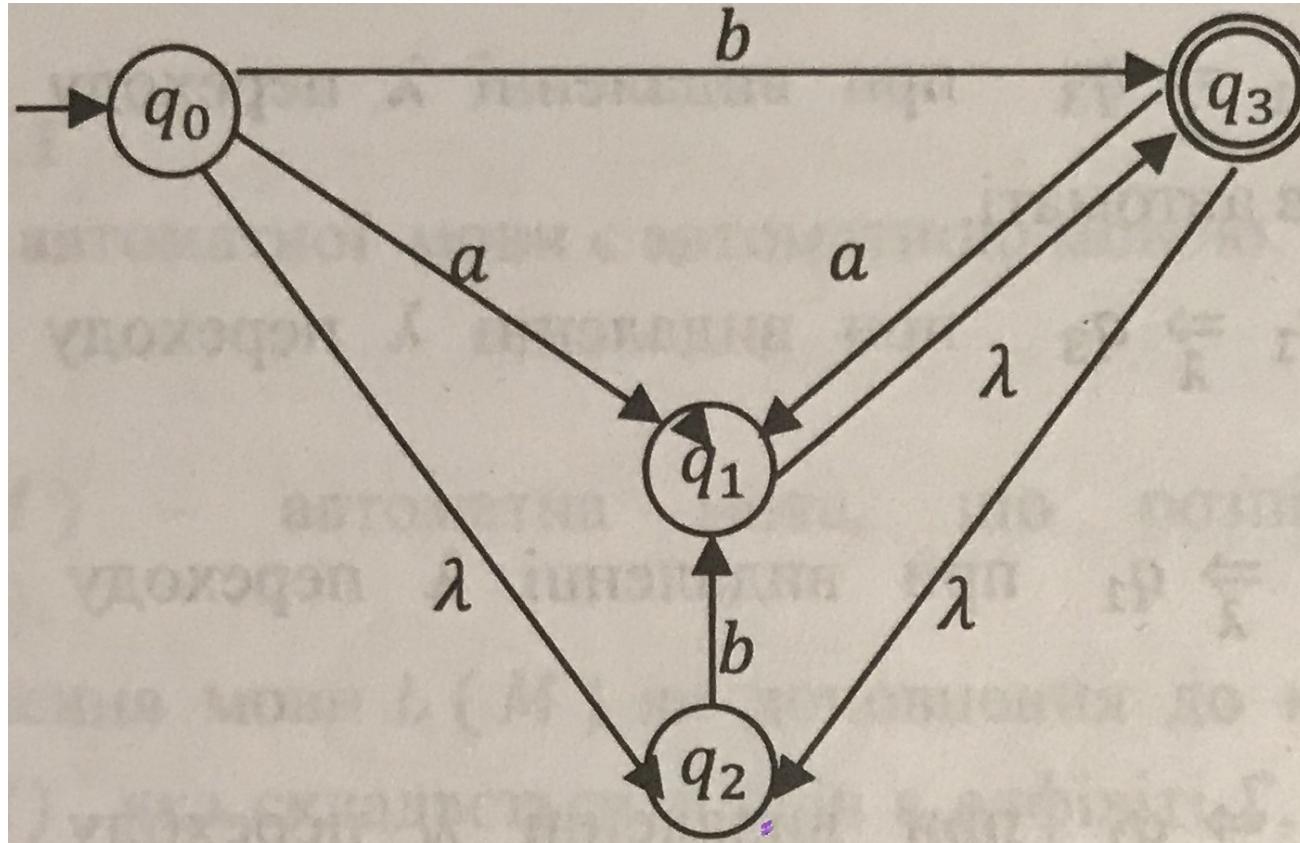
Якщо ж $q \in Q'$ (тобто крім дуги з міткою λ до нього входить принаймні одна дуга з міткою $b \in \Sigma, b \neq \lambda$), природно, в M' збережеться дуга $p \xrightarrow{a} r$ і символ a буде одним з символів, що належать мітці цієї дуги.

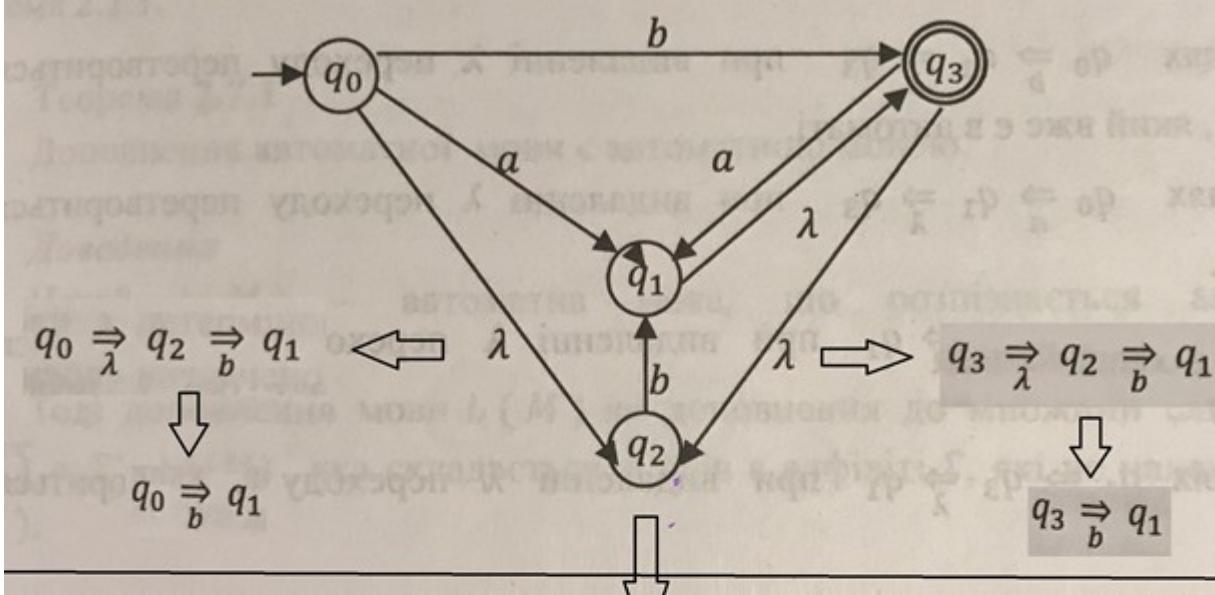


3. Множина заключних станів F' скінченного автомата M' містить всі стани $q \in Q'$, які або належали до заключних станів початкового автомата M , або з яких веде шлях ненульової довжини з q в заключний стан $f \in F$ початкового автомата M з міткою шляху λ

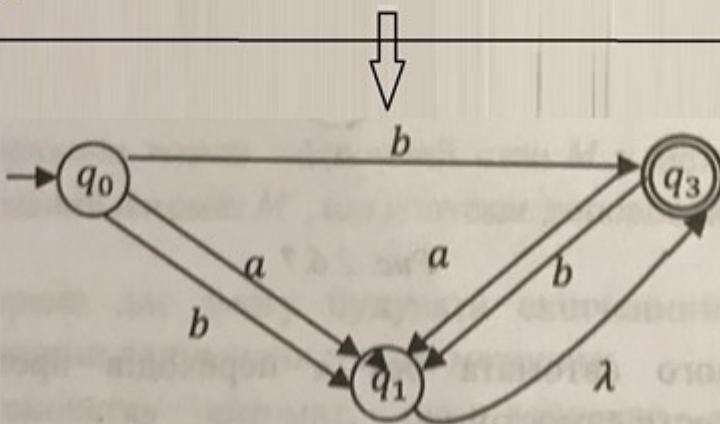


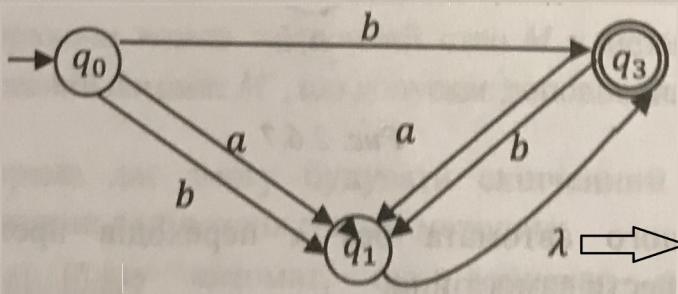
Видалення λ -переходів. Приклад:





q_2 стає **непродуктивним** станом
(q_2 також стає **недосяжним** станом)

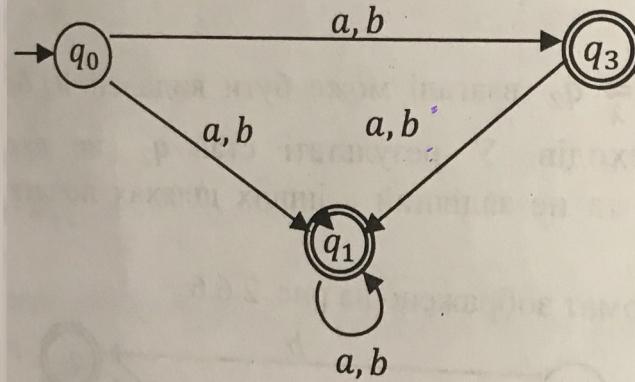




$$\begin{aligned}
 q_0 \xrightarrow{b} q_1 \xrightarrow{\lambda} q_3 &\Rightarrow q_0 \xrightarrow{b} q_3 \\
 q_0 \xrightarrow{a} q_1 \xrightarrow{\lambda} q_3 &\Rightarrow q_0 \xrightarrow{a} q_3 \\
 q_1 \xrightarrow{\lambda} q_3 \xrightarrow{b} q_1 &\Rightarrow q_1 \xrightarrow{b} q_1 \\
 q_1 \xrightarrow{\lambda} q_3 \xrightarrow{a} q_1 &\Rightarrow q_1 \xrightarrow{a} q_1
 \end{aligned}$$



Оскільки є λ перехід $q_1 \xrightarrow{\lambda} q_3$ і q_3 – заключний стан, то при його видаленні q_1 стає теж заключним.



Скінчений автомат $M = \langle Q, \Sigma, \Delta, I, F \rangle$ називається **детермінованим** (deterministic), якщо

1. Множина I містить рівно один елемент;

2. Для кожного переходу $\langle p, x, q \rangle \in \Delta$ виконується рівність $|x| = 1$, тобто мітки переходів автомата є однобуквеними;

3. Автомат не містить дуг з пустими λ мітками;

4. Для кожного символу $a \in \Sigma$ і для довільного стану $p \in Q$ існує тільки один стан $q \in Q$ такий, що $\langle p, a, q \rangle \in \Delta$, тобто перехід зі стану p у стан q по дузі з міткою a повинен бути єдиним дляожної букви алфавиту.

Детермінізація

Вважаємо, що автомат, який підлягає процедурі власне детермінізації, вже не містить λ переходів.

Спочатку зауважимо, що автоматна граматика (надалі А-граматика) відповідає детермінованому автомату тоді і тільки тоді, коли за кожним її правилом вигляду

$$A \rightarrow a_1 A_1 | \dots | a_n A_n \quad \text{або} \quad A \rightarrow a_1 A_1 | \dots | a_n A_n | \epsilon$$

виконується умова: $a_i \neq a_j$ при $i \neq j$.

Для кожного правила, ліві частини яких є нетерміналами з множини $\{A_1, \dots, A_n\}$, а праві частини цих правил розпочинаються однаковими термінальними символами, введемо новий нетермінальний символ, який позначимо як $[A_1 \dots A_n]$ і визначимо для нього правило граматики так, щоб його права частина складалася з *правих частин правил для кожного A_i , що входить в множину $\{A_1, \dots, A_n\}$* , тобто задамо правило

$$[A_1 \dots A_n] \rightarrow r_1 | \dots | r_n ,$$

де r_i – права частина правила для A_i , тобто $A_i \rightarrow r_i$.

Тепер побудуємо граматику $G' = (N', T, P', S)$, де N' отримується з N додаванням визначених вище нових нетермінальних символів, а P' отримано додаванням до P правил для нових нетермінальних символів: в кожному правилі всі члени вигляду aA_1, \dots, aA_n з одним і тим же терміналом $a \in T$ заміняються одним правилом $a[A_1, \dots, A_n]$.

Таким правилам відповідає детермінований автомат, еквівалентний недетермінованому, оскільки $a[A_1, \dots, A_n] = aA_1 | \dots | aA_n$.

Оскільки не всі нетермінальні символи $[A_1, \dots, A_n]$ досяжні з S , то для побудови P' достатньо використовувати тільки необхідні правила, починаючи з правила для S .

Отриманий детермінований скінчений автомат повинен далі обов'язково пройти процедуру видалення непродуктивних та недосяжних станів.

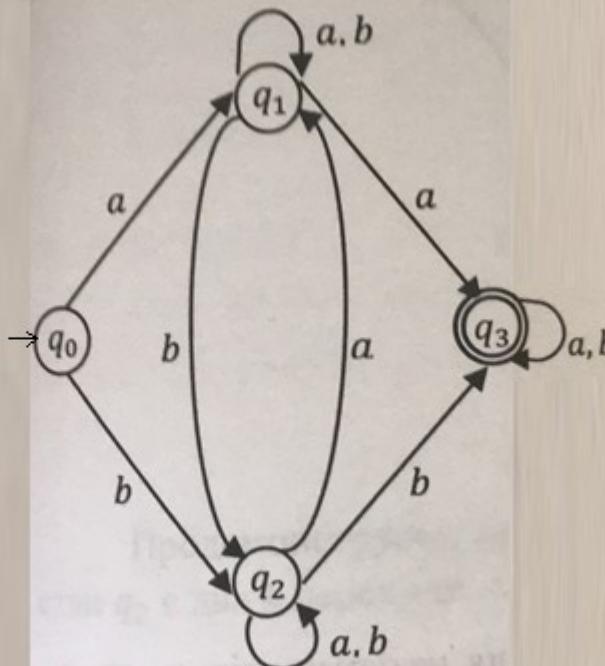
Детермінізація недетермінованого скінченного автомата. Приклад:

Недетермінований автомат

$M_d = \langle Q, \Sigma, \Delta, I, F \rangle$, де $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, $I = \{q_0\}$
 $F = \{q_3\}$,

$\Delta = \{ \langle q_0, a, q_1 \rangle, \langle q_0, b, q_2 \rangle, \langle q_1, a, q_1 \rangle, \langle q_1, b, q_1 \rangle, \langle q_2, a, q_2 \rangle, \langle q_2, b, q_2 \rangle,$
 $\langle q_1, b, q_2 \rangle, \langle q_2, a, q_1 \rangle, \langle q_1, a, q_3 \rangle, \langle q_2, b, q_3 \rangle, \langle q_3, a, q_3 \rangle, \langle q_3, b, q_3 \rangle,$

не містить λ переходів, тому до нього
можна одразу застосувати власне детермінізацію.



Побудуємо еквівалентну цьому автомату граматику з правилами

$$q_0 \rightarrow aq_1 | bq_2,$$

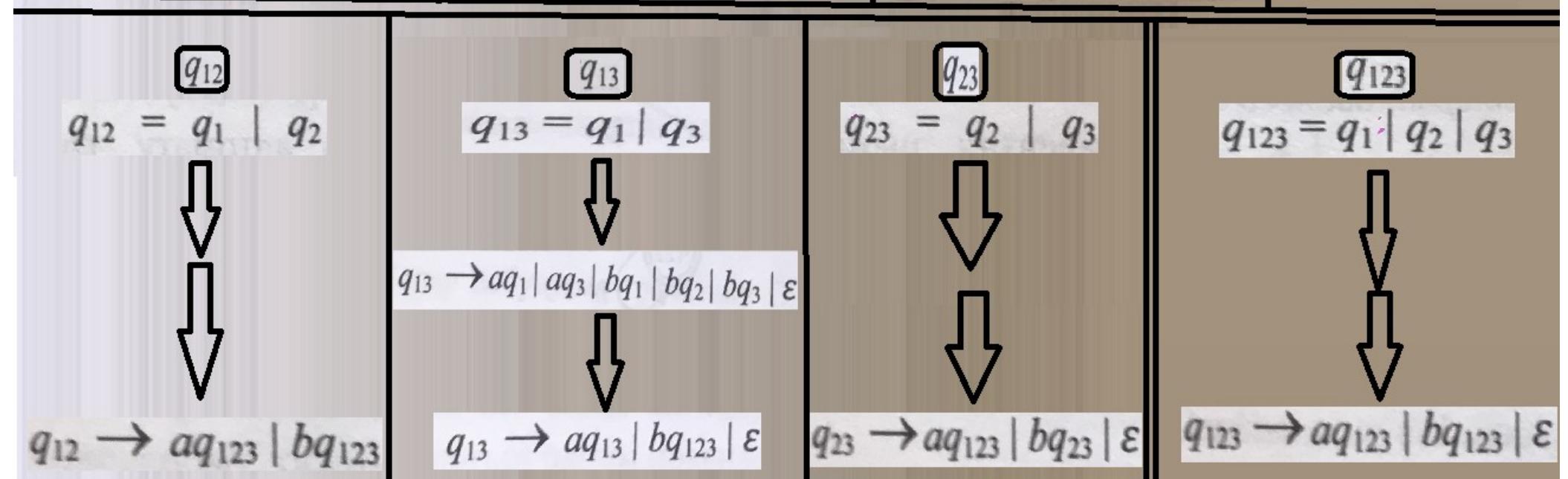
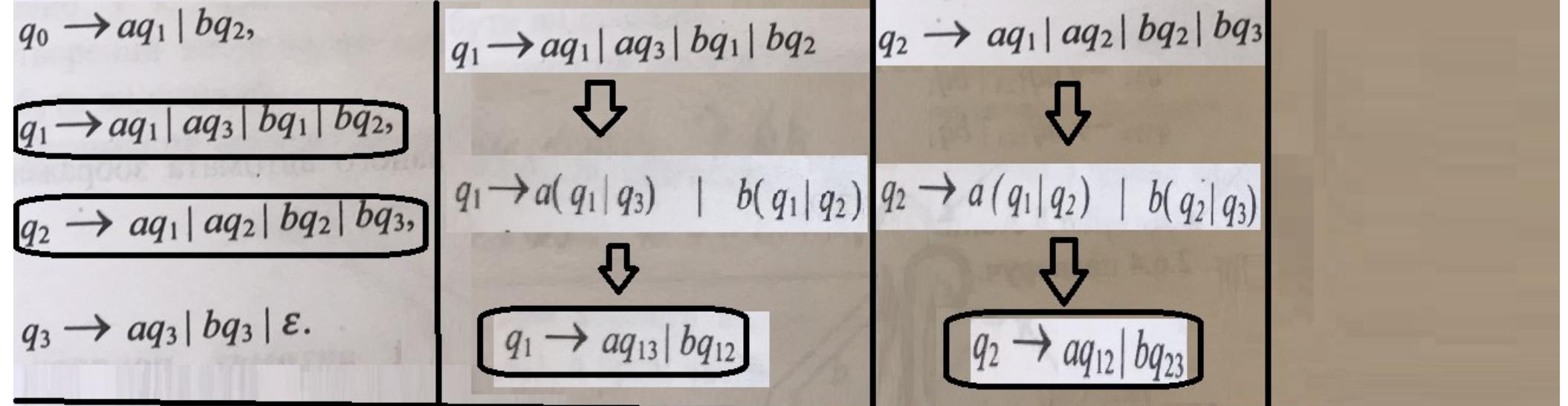
$$q_1 \rightarrow aq_1 | aq_3 | bq_1 | bq_2,$$

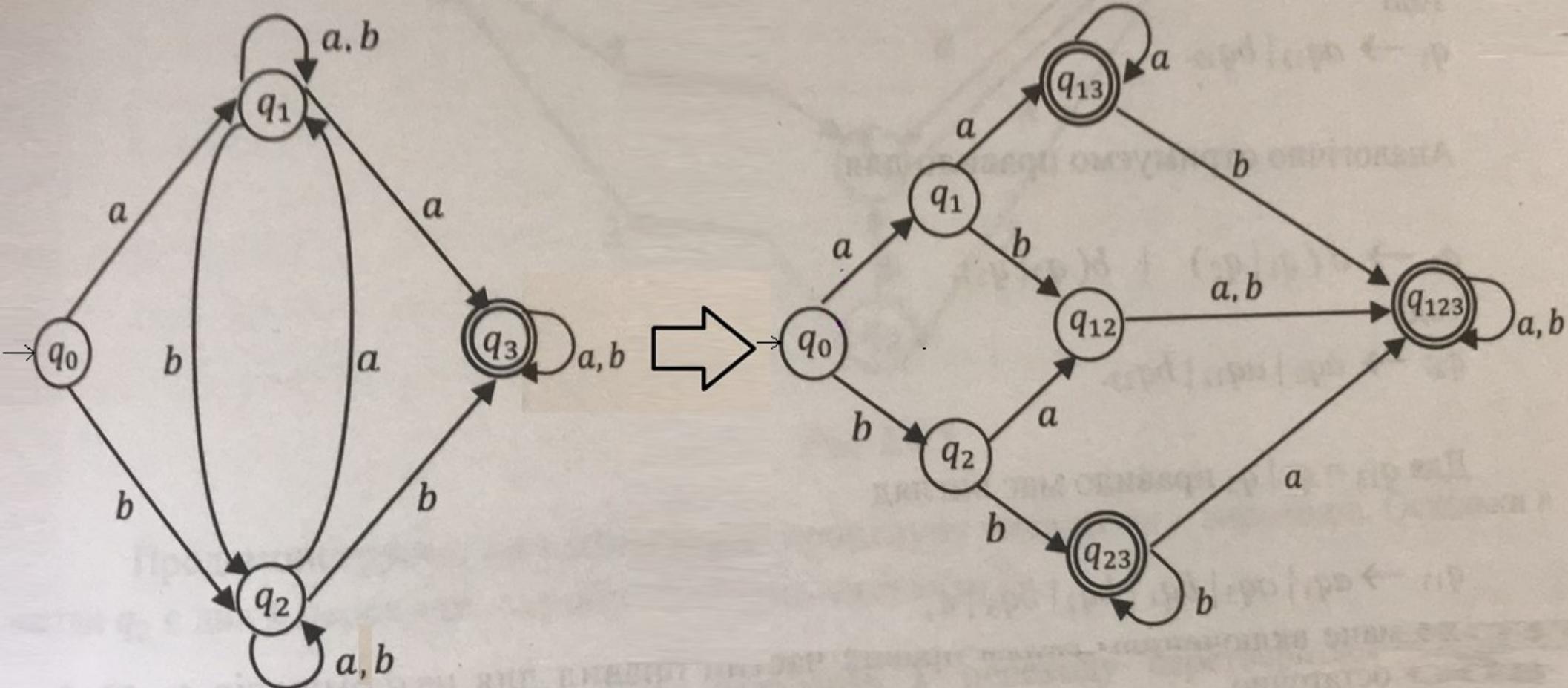
$$q_2 \rightarrow aq_1 | aq_2 | bq_2 | bq_3,$$

$$q_3 \rightarrow aq_3 | bq_3 | \varepsilon.$$

$S \rightarrow aA bB,$	$S \rightarrow aA,$ $S \rightarrow bB,$
$A \rightarrow aA aF bA bB,$	$A \rightarrow aA,$ $A \rightarrow aF,$ $A \rightarrow bA,$ $A \rightarrow bB,$
$B \rightarrow aA aB bB bF,$	$B \rightarrow aA,$ $B \rightarrow aB,$ $B \rightarrow bB,$ $B \rightarrow bF,$
$F \rightarrow aF bF \varepsilon.$	$F \rightarrow aF,$ $F \rightarrow bF,$ $F \rightarrow \varepsilon.$

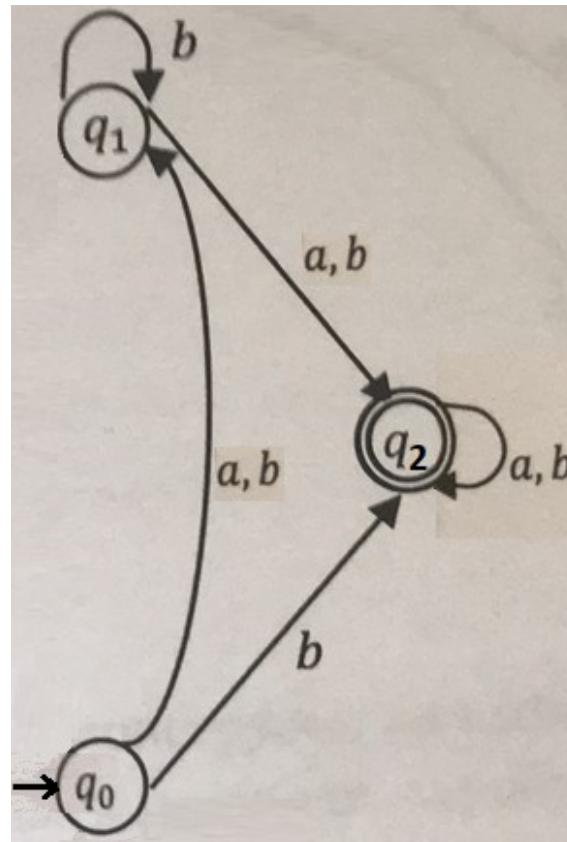
Правило $q_3 \rightarrow \varepsilon$ додається в граматику, оскільки q_3 є заключним станом.





Контрольне завдання №13

Виконати детермінізацію заданого недетермінованого скінченного автомата.



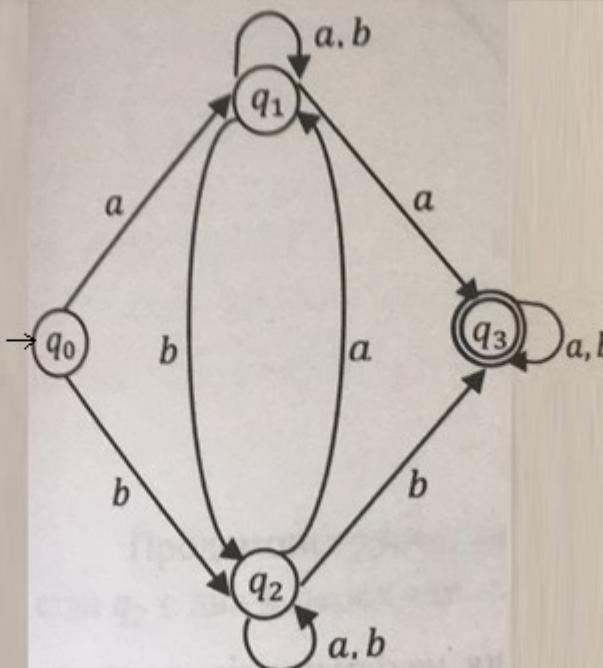
Детермінізація недетермінованого скінченного автомату. Приклад 2:

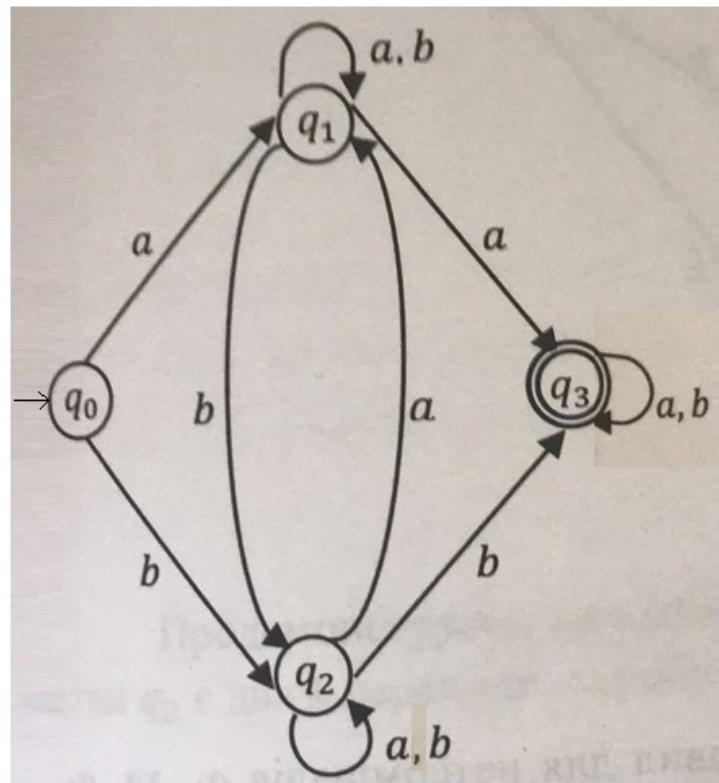
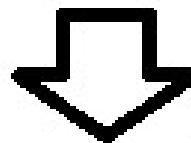
Недетермінований автомат

$M_d = \langle Q, \Sigma, \Delta, I, F \rangle$, де $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, $I = \{q_0\}$
 $F = \{q_3\}$,

$\Delta = \{ \langle q_0, a, q_1 \rangle, \langle q_0, b, q_2 \rangle, \langle q_1, a, q_1 \rangle, \langle q_1, b, q_1 \rangle, \langle q_2, a, q_2 \rangle, \langle q_2, b, q_2 \rangle,$
 $\langle q_1, b, q_2 \rangle, \langle q_2, a, q_1 \rangle, \langle q_1, a, q_3 \rangle, \langle q_2, b, q_3 \rangle, \langle q_3, a, q_3 \rangle, \langle q_3, b, q_3 \rangle,$

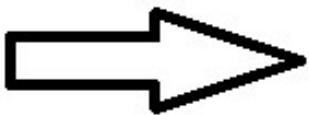
не містить λ переходів, тому до нього
можна одразу застосувати власне детермінізацію.



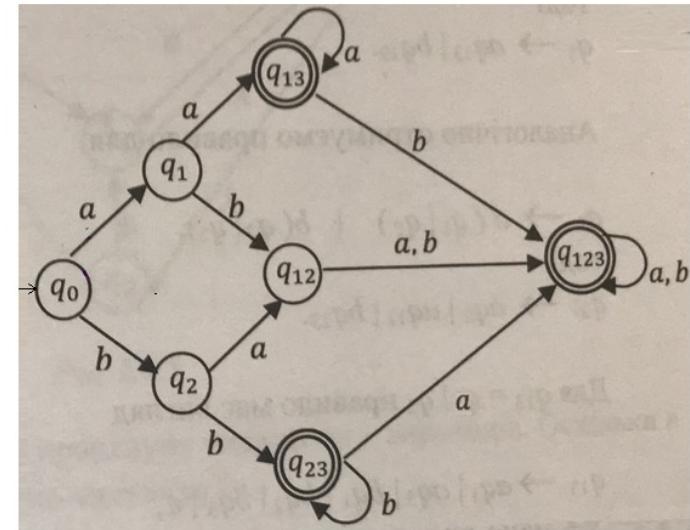
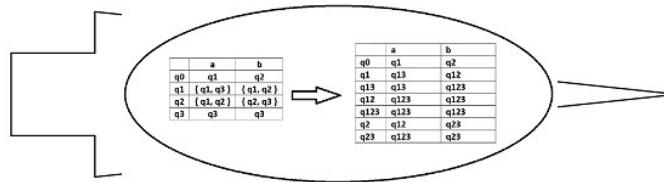
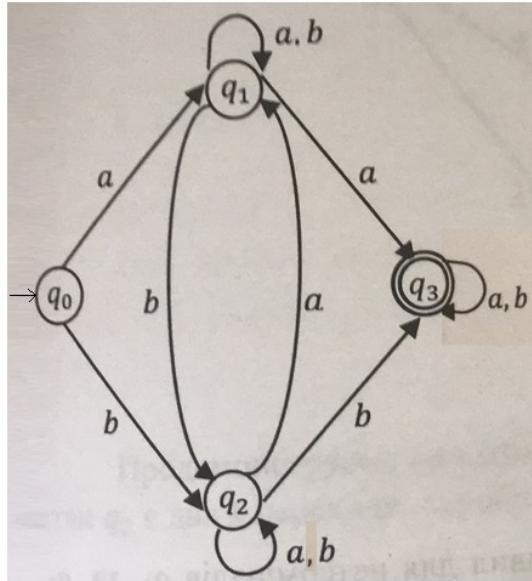
$$\Delta = \{ \langle q_0, a, q_1 \rangle, \langle q_0, b, q_2 \rangle, \langle q_1, a, q_1 \rangle, \langle q_1, b, q_1 \rangle, \langle q_2, a, q_2 \rangle, \langle q_2, b, q_2 \rangle, \langle q_1, b, q_2 \rangle, \langle q_2, a, q_1 \rangle, \langle q_1, a, q_3 \rangle, \langle q_2, b, q_3 \rangle, \langle q_3, a, q_3 \rangle, \langle q_3, b, q_3 \rangle,$$


	a	b
q0	q1	q2
q1	{ q1, q3 }	{ q1, q2 }
q2	{ q1, q2 }	{ q2, q3 }
q3	q3	q3

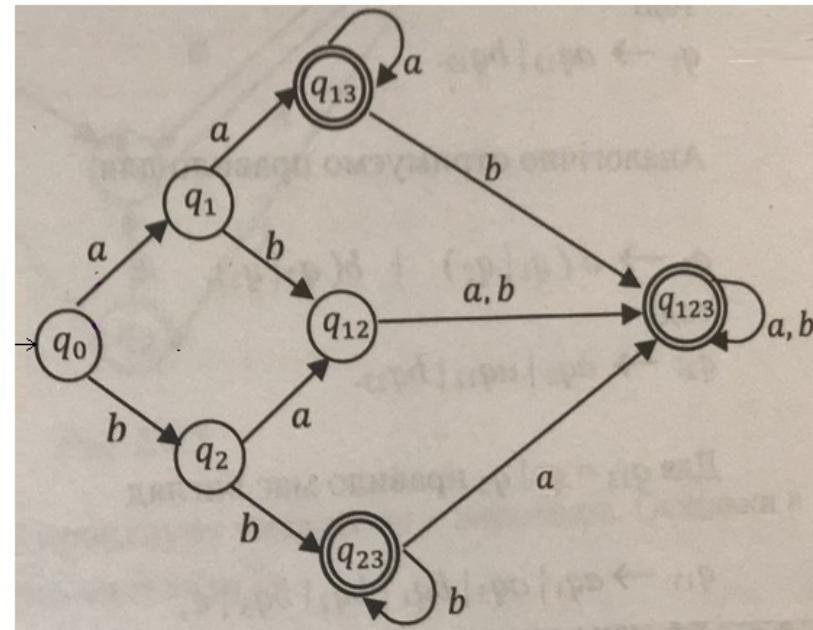
	a	b
q0	q1	q2
q1	{ q1, q3 }	{ q1, q2 }
q2	{ q1, q2 }	{ q2, q3 }
q3	q3	q3



	a	b
q0	q1	q2
q1	q13	q12
q13	q13	q123
q12	q123	q123
q123	q123	q123
q2	q12	q23
q23	q123	q23



	a	b
q0	q1	q2
q1	q13	q12
q13	q13	q123
q12	q123	q123
q123	q123	q123
q2	q12	q23
q23	q123	q23



Контрольне завдання №14

Виконати детермінізацію заданого недетермінованого скінченного автомата (*автомат з контрольного завдання №13*) шляхом формування таблиці переходів його детермінованої версії на основі таблиці переходів недетермінованої версії.

