Формални Езици и Езикови Процесори
ТУ, кат. КС, летен семестър 2012

Лекция 5

Тема:

Разпознаватели
(Крайни Автомати)

Съдържание:

- Релация граматика, език, разпознавател
- Крайни Автомати /КА/
  - Въведение
  - Граф на преходите
  - Таблица на преходите
  - Обобщени алгоритми на действие
- Демо програми

От КСГ към друг тип граматики

- Задача: Да се съставят КСГ продукции, описващи следния език на тройни симетрични низове

$$L = \{ a^n b^n c^n \mid_{n>=1} \}$$

5 мин за решение, ако е възможно.

Едно решение извън КСГ е показано на следващия слайд.

От КСГ към друг тип граматики

Решението:

$S \rightarrow a\ S\ B\ C \mid a\ b\ C$

$C\ B \rightarrow B\ C$

$b\ B \rightarrow b\ b$

$b\ C \rightarrow b\ c$

$c\ C \rightarrow c\ c$

Класификация на граматиките

Формални граматики според Хомски /Chomsky/

Class 0	Граматики с фразова структура (Без ограничения в правилата) Общ формат: $\alpha \rightarrow \beta$
Class 1	Контекстно-зависими граматики Общ формат: $u1\ X\ u2 \rightarrow u1\ \alpha\ u2$
Class 2	Контекстно-свободни граматики Общ формат: $X \rightarrow \alpha$
Class 3	Линейни граматики Общ формат: $X \rightarrow a$ $X \rightarrow Y\ a$

ЛГ като частен случай на КСГ

- Граматиката на идентификаторите бе въведена в миналата лекция като КСГ, но по същество тя е ЛГ.

<Ident> ::= Letter | <Ident> Letter | <Ident> Digit

или

$Id \rightarrow L \mid Id\ L \mid Id\ D$

ЛГ като частен случай на КСГ

- Граматиката на целите десетични константи без знак бе въведена в миналата лекция като КСГ, но по същество тя е ЛГ.

<Const> ::= Digit | <Const> Digit

Релация Граматика - Език

1. Един формален език може да се опише с повече от една граматика /еквивалентни граматики/.

Пример: езикът на симетричните низове се описва с три различни граматики.

Релация Граматика - Език

1. Един формален език може да се опише с повече от една граматика /еквивалентни граматики/.

   Пример: езикът на симетричните низове се описва с три различни граматики.

1. Всяка граматика описва само един език.

Граматиките на Хомски

Class 0 /ГФС/ представя универсалното множество на всички граматики (езици).

Граматиките на Хомски

Class 0 /ГФС/ представя универсалното множество на всички граматики (езици).

Class 1 /КЗГ/ е подмножество на class 0.

Граматиките на Хомски

Class 0 /ГФС/ представя универсалното множество на всички граматики (езици).

Class 1 /КЗГ/ е подмножество на class 0.

Class 2 /КСГ/ е подмножество на class 1.

Граматиките на Хомски

Class 0 /ГФС/ представя универсалното множество на всички граматики (езици).

Class 1 /КЗГ/ е подмножество на class 0.

Class 2 /КСГ/ е подмножество на class 1.

Class 3 /ЛГ/ е подмножество на class 2.

Къде са РЕ?

Отговорът:

Регулярни Езици /РИ/ ≡

Линейни Езици /ЛГ/ ≡

Автоматни Езици /КА/

Припомняне

Един формален език може да се опише:

- По синтетичен път
  - На основата на граматика като средство за генериране на низове
- По аналитичен път
  - На основата на разпознавател на низове

Автоматни Граматики/Езици

- Един език е автоматен, ако съществува краен автомат /КА - FS automata, FS machine, FS recognizer, FS acceptor/, който за всеки низ след сканирането му, може да даде заключение като го приеме за изречение от езика или го отхвърли като грешен низ.
- Граматика, която поражда автоматен език, се нарича автоматна граматика.

Езици и Автомати

Въведение в разпознаватели.

Обща схема (структура):

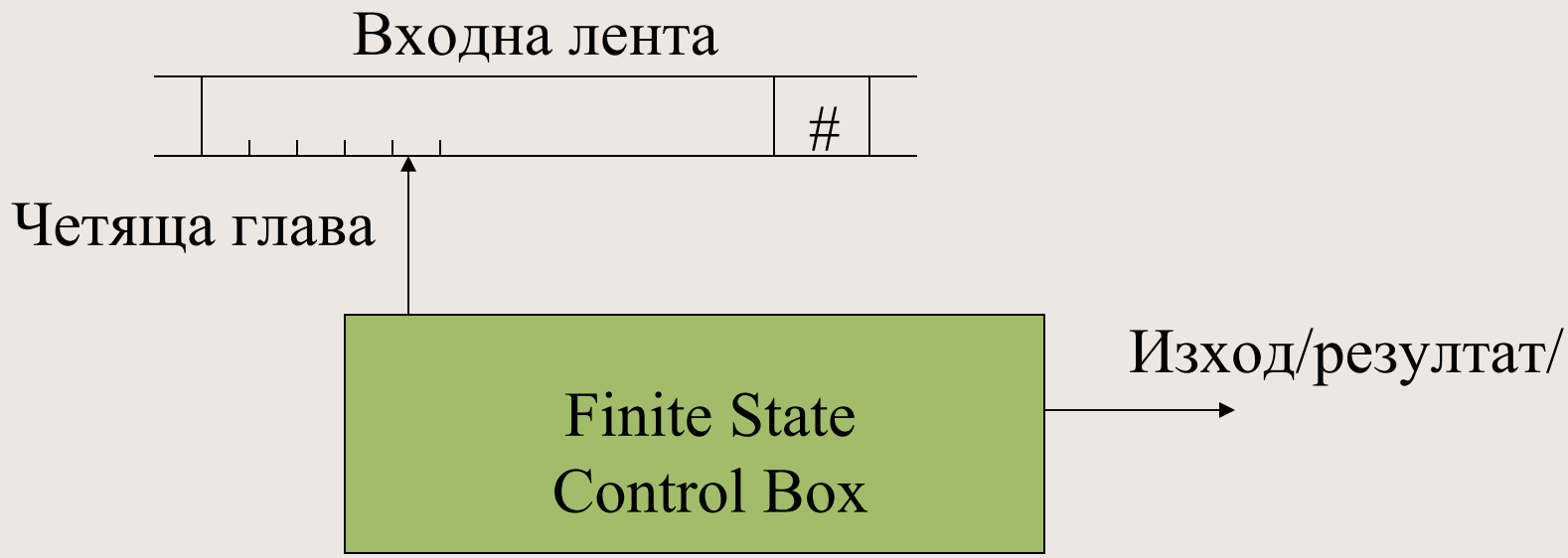
Входна лента (съдържа низ за сканиране);

Сканиращ показалец – четяща глава;

Упр. У-во с краен брой състояния;

Изход - /Резултат/

Най- обща схема на КА



Формална дефиниция на КА

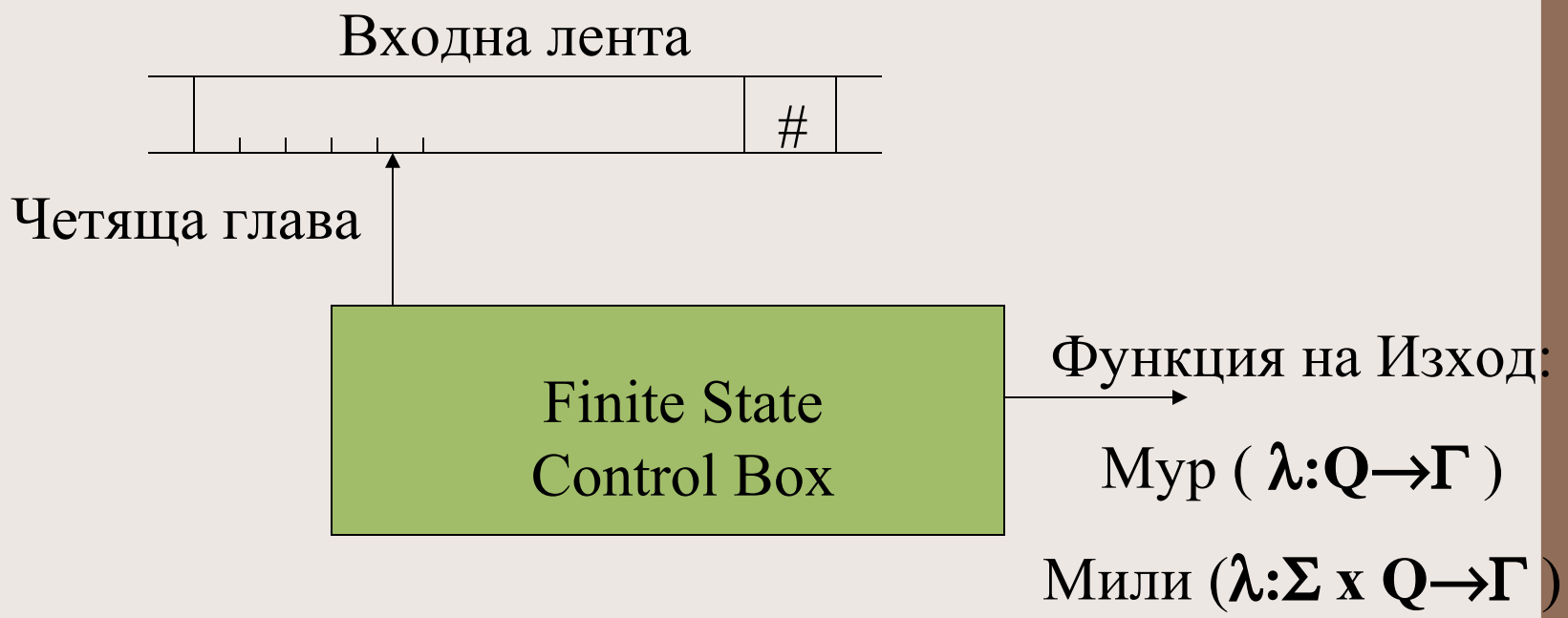
Два подхода:

FSA = (Σ, Q, Γ, δ, λ)	**FSA = (Σ, Q, δ, q_0, Q_f)**
Σ – входна азбука	**Σ – входна азбука**
Q – м-во вътрешни състояния	**Q – м-во вътрешни състояния**
Γ – азбука на изхода	**δ – функция на прехода: Σ x Q → Q**
δ – функция на прехода: Σ x Q → Q	**q_0 – начално вътрешно състояние**
λ – функция на изхода: Σ x Q→Γ или Q→Γ	**Q_f – м-во от заключителни състояния**

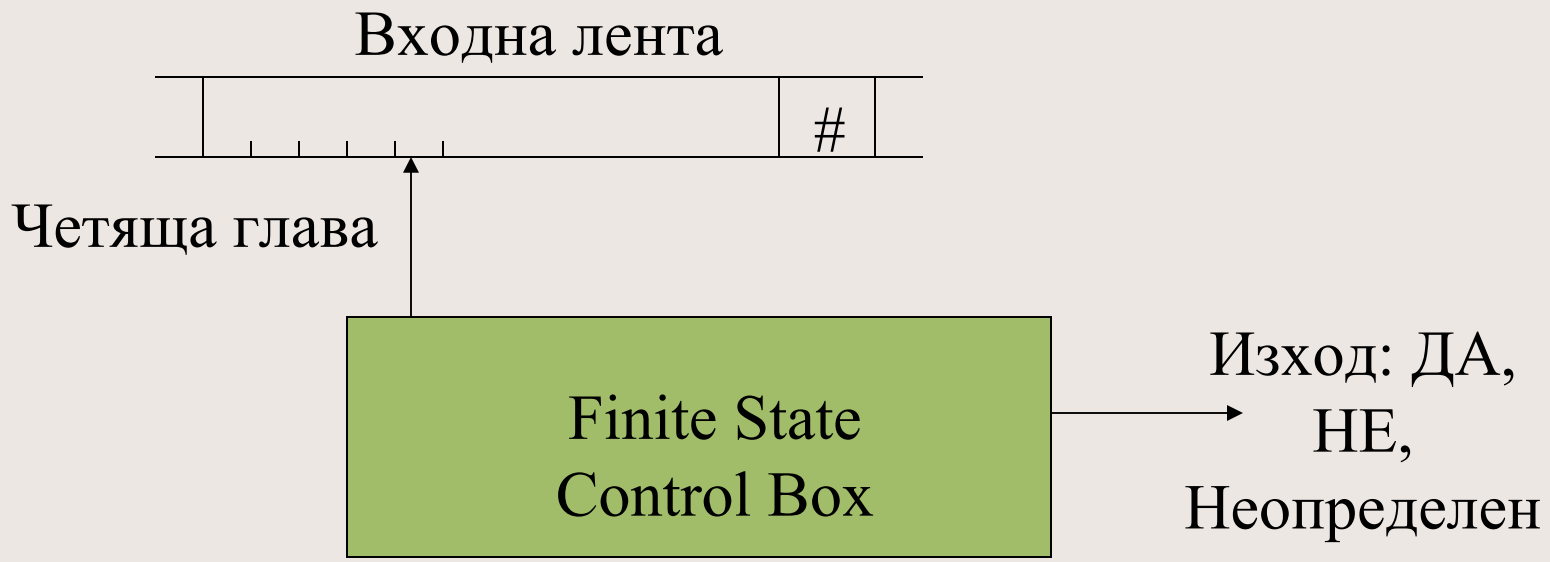
Класификация на КА

- Видове КА /различават се по структурата на изхода:
  - Преобразуватели /transducer/
    - Изход: Ф-ия на изходите /Мур, Мили/
  - Разпознаватели /recognizer, acceptor/
    - Изход: Да, Не, Неопределен

КА като Преобразувател



КА като Разпознавател



Разпознавател(и)

Синоними на краен автомат /КА/:

Finite Automata (FA) ≡

Finite State Acceptor (FSA) ≡

Finite State Engine (FSE) ≡

Finite State Machine (FSM) ≡

Finite State Recognizer (FSR)

Формална дефиниция на КА

Два подхода:

FSA = (Σ, Q, Γ, δ, λ)	**FSA = (Σ, Q, δ, q_0, Q_f)**
Σ – входна азбука	**Σ – входна азбука**
Q – м-во вътрешни състояния	**Q – м-во вътрешни състояния**
Γ – азбука на изхода	**δ – функция на прехода: Σ x Q → Q**
δ – функция на прехода: Σ x Q → Q	**q_0 – начално вътрешно състояние**
λ – функция на изхода: Σ x Q→Γ or Q→Γ	**Q_f – м-во от заключителни състояния**

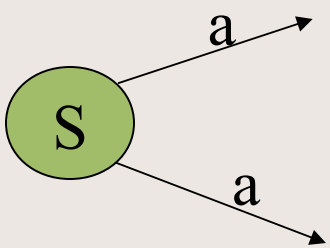
КА – друга класификация

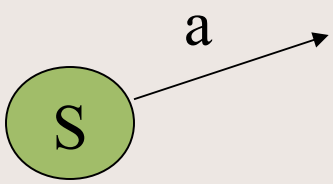
- Не детерминирани КА (NFA)

  От състояние **s** под управление на входен символ **a** има две или повече дъги, именовани с входен символ **a** и излизащи от **s**.

- Детерминирани КА (DFA)

  за всяко състояние **s** и входен символ **a** има не повече от една дъга, именована с входен символ **a** и излизаща от **s**.


a
S
a


a
S

Разпознавател(и) Recognizer(s), Acceptor(s)

- Разпознавател е програма, която чете/сканира/ входен низ **x** и връща “YES”, ако **x** е изречение от езика или връща “NO” в противен случай.
- От РИ може да се получи разпознавател като се построи КА със съответна таблица на преходите.

Recognizer(s), Acceptor(s)

- Два вида КА
  - DFA
  - NFA
- И двата КА /DFA и NFA/ разпознават прецизно регулярни множества от символни низове. Следователно, и DFA и NFA могат да разпознават низове, описвани с РИ.
- DFA са по-бързи разпознаватели от NFA.
- DFA заемат повече памет от еквивалентни NFA.

Два начина за представяне на КА

Чрез граф на преходите

- Възлите се именоват с вътрешни състояния.
- Дъгите свързват възли и представят функцията на преходите.

Чрез таблица /матрица/ на преходите

- редове се именоват с входни символи.
- колони се именоват с вътрешни състояния.
- в таблицата се записват вътрешни състояния.

Двата начина са еквивалентни и взаимно свързани.

Примери за DFA и NFA

Пример за DFA

Дадена е таблица на преходите. Да се построи съответен граф на преходите

Входна азбука $\Sigma = \{ a, b, c \}$

М-во състояния $Q = \{ q1, q2, q3, q4 \}$

Изходна азбука $\Gamma = \{ 0, 1, 2 \}$

Примери за DFA и NFA

Пример за NFA низове

$L=\{ a^m b^n \mid_{m,n>=1} \}$ РИ = $a^+ b^+$

Даден е граф на преходите. Да се построи таблицата на преходите

Входна азбука $\Sigma = \{ a, b \}$

Начално състояние = q0

М-во състояния $Q = \{ q1 \}$

Крайно състояние = q2

Примери за DFA и NFA

Пример за NFA за низове от език, описван с РИ = $a.a^* \mid b.b^* = a^+ \mid b^+$

Входна азбука $\Sigma = \{ a, b \}$

Начално състояние = q0

М-во състояния $Q = \{ q1, q3 \}$

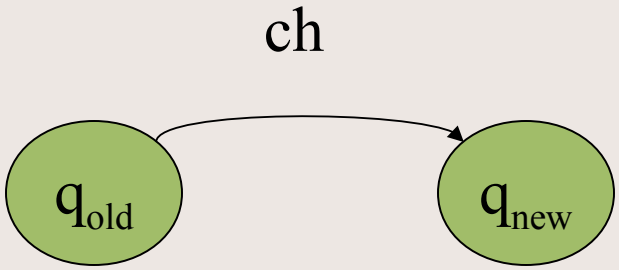
М-во крайни състояния = { q2, q4 }

FSA симулатор - алгоритъм

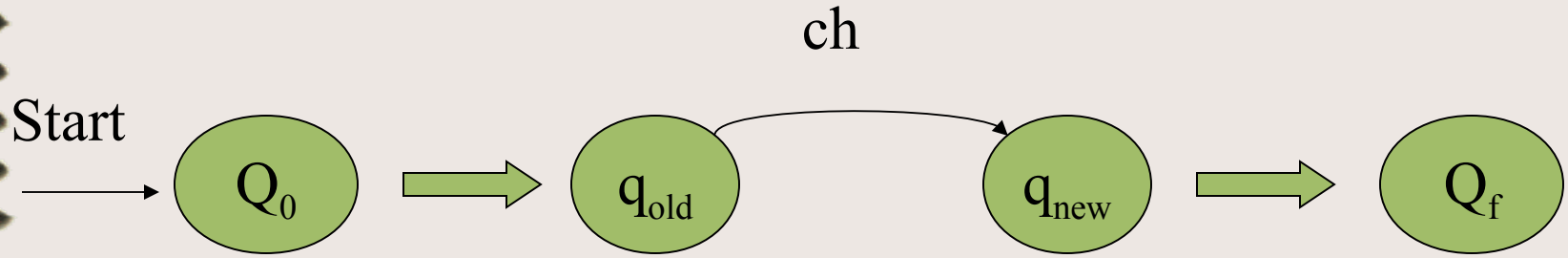
КА разпознава входен низ ***x***, ако и само ако има път в графа на преходите от начално състояние q0 до финално състояние (в случай на DFA) или до някое от финалните състояния (в случай на NFA) такъв, че имената на дъгите, които го съставят, образуват конфигурацията на входния низ ***x***.

Път /пътека/ се представя като последователност от преходи между състояния, наречени **move(s)**.

Единичен преход /move/



Път/пътека/ от преходи /moves/



state move(state q, symbol ch);

Въвежда се процедура, която симулира преход от старо/текущо/ състояние **q** в ново/актуално/ състояние под въздействие на входен символ **ch**.

Прототип:

```
state move(state q, symbol ch);
```

Други подробности

Функция, която чете и връща поредния /текущия/ входен символ:

Прототип:

```
symbol newchar();
```

Две ПП

- Функция връща ново състояние след единичен преход под въздействие на текущия входен символ:

```
state move(state q, symbol ch);
```

- Функция връща поредния входен символ:

```
symbol newchar();
```

Спецификация на алгоритъма

Цел: симулатор на КА, оперирaщ като разпознавател на низове;

Вход: входен низ, ограничен с EOF

КА с начално състояние q_0 и

множество заключителни състояния Q_f.

Изход: YES, ако КА разпознава входен низ или NO в противен случай.

Прототип на функция симулатор на КА разпознавател:

```
answer FSA_Recognizer();
```

Дефиниция на функция **FSA_Recognizer()**

```
answer FSA_Recognizer()
 {
   state q; symbol ch;  // локални променливи
   q = q0;              // начална стойност за q и ch
   ch = newchar();
   // цикъл до изчерпване на входния поток
   while ( ch !=  EOF )  do
       q = move(q, ch);
       ch = newchar();
   endo;
   if ( q in Qf ) return "YES";
       else   return "NO";
 }
```

Прототип на функция симулатор на КА преобразувател:

```
void FSA_Transducer();
```

Дефиниция на функция **FSA_Transducer()**

```
void FSA_Transducer()
 {
   state q; symbol ch, outch;  // локални pr-иви
   q = q0;                     // начална стойност за q и ch
   ch = newchar();
   // цикъл до изчерпване на входния поток
   while ( ch !=  EOF )  do
       outch = convert(ch, q); // Mealey
       outch = convert(q);     // Moore
       q = move(q, ch);
       ch = newchar();
   endo;
 }
```

Демо програми

КА симулатор	C/C++	BASIC
Разпозанавател на идентификатори	fsaid1.exe, fsaid1m.exe	ka1stbo.bas (gwbasic.exe)
Разпознавател на константи	fsacns1.exe, fsacns1m.exe	karistbo.bas (gwbasic.exe)

Благодаря
За
Вниманието