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

Лекция 1-2

Тема:

Въведение в ЕП

Увод в ТФЕ: Азбука, Низ, Език

Съдържание:

- Йерархия на ПЕ
- ЕП и мета програми, видове ЕП
- Алгоритъм на К. и Алгоритъм на И.
- УНКОЛ (UNCOL) и преносимост на ПО
- Азбука, Низ, Език
- Операции над множества от символни низове

Част 1

Въведение в Езикови Процесори

Йерархия на ПЕ

- Компютър
  - Машинен език
  - Асемблерен език
  - ПЕ от Високо Ниво
  - Естествен език
- Потребител

Формат на инструкции

- Машинен Език
  - Последователност от нули и единици (0 и 1)
  - Структура на МЕ инструкция:

<КодОп> <операнд(и)*>

КодОп, следван от 0, 1 или повече операнди - адреси или явни стойности

- 0010 0000 0000 0100
- 1000 0000 0000 0101
- 0011 0000 0000 0110

Формат на инструкции

Асемблерен език

- Мнемоничен КодОп
- Символни имена означават операнди, а също и непосредствени операнди
  - MOV AX, PRICE
  - ADD AX, TAX
  - STO AX, COST

MOV AX, PRICE
ADD AX, #20
STO AX, COST

Формат на инструкции

- ПЕВН (C, C++, C#, VB, Java)
  - Пример на оператор за присвояване
    - cost = price + tax;
- Естествен език
  - Add contents of *Price* with *Tax*,
  - Save result to *Cost.*

Потребността от Езикови Процесори

Една програма (АЕ или ПЕВН) следва да се
транслира (to compile or to interpret),
или преобразува,
или конвертира
До съответната МЕ форма

ЕП и мета програми

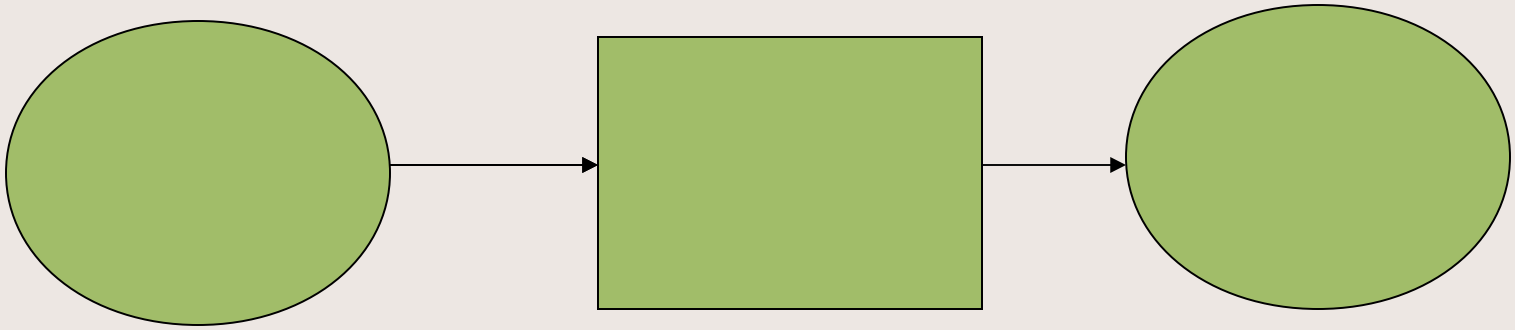
- Мета език, мета символ,
- Мета програма,
- Мета клас
- ЕП са мета програми
- Видове ЕП
  - Според входната програма
    - Транслатор, асемблер, конвертор
    - Силициев компилатор, интерпретатор на заявки БД
  - Според начина на работа
    - Компилатор
    - Интерпретатор

ЕП са мета програми

Първична програма

Езиков процесор

Обектна програма



Видове ЕП

Първична програма		Езиков процесор		Обектна програма
ЕВН	→	Транслатор	→	МЕ
Асембл.ез.	→	Асемблер	→	МЕ
ЕВН1	→	Конвертор	→	ЕВН2
Език1	→	Пре(макро)П.	→	Език1

Специфични ЕП

- Silicon compiler – Силициев компилатор, входният език е стандартен, но идентификаторите именоват не клетки от паметта, а сигнали от електрическа схема. Изходът е проектирана ел верига в термините на подходящ специализиран език, например
- VHDL

Специфични ЕП

- Query interpreter – Интерпретатор на заявки. Входът е предикат с лог. И рел. операции, който се превежда в команди за сканиране и търсене в БД на записи, които удовлетворяват входния предикат. Например
- QBE, SQL

Видове транслатори

- Компилатори
- Интерпретатори

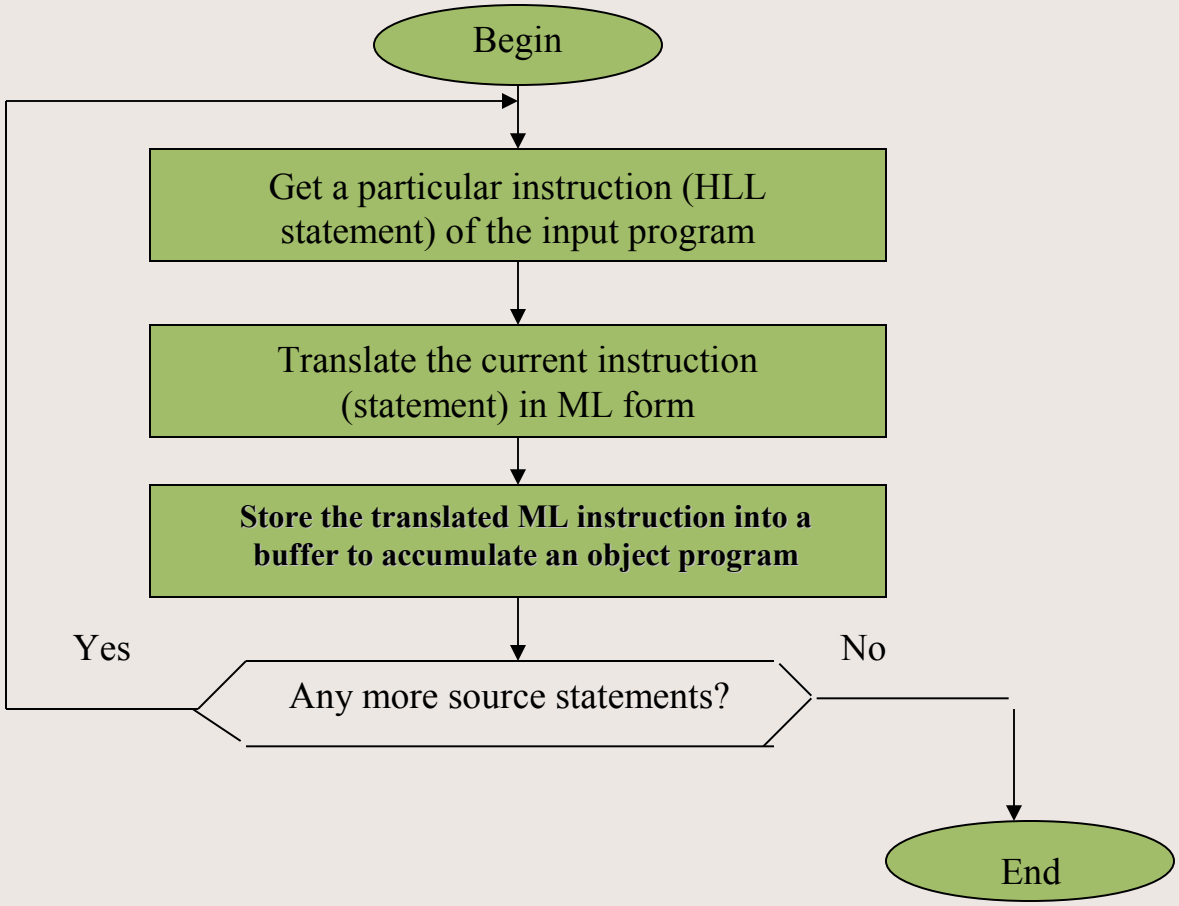
Алгоритъм на К. и Алгоритъм на И.

- Блок схема на обобщен алгоритъм на компилатор
- Блок схема на обобщен алгоритъм на интерпретатор

Обобщен алгоритъм на компилатор

1. Начало.
2. Четене на EBH инструкция (HLL statement) от вх. Програма.
3. Превод на текущата инструкция в ME формат.
4. **Буфериране на превода за формиране на обектна програма.**
5. Има ли още инструкции на входа? Ако Да, премини към 2. Иначе, премини към 6.
6. Край.

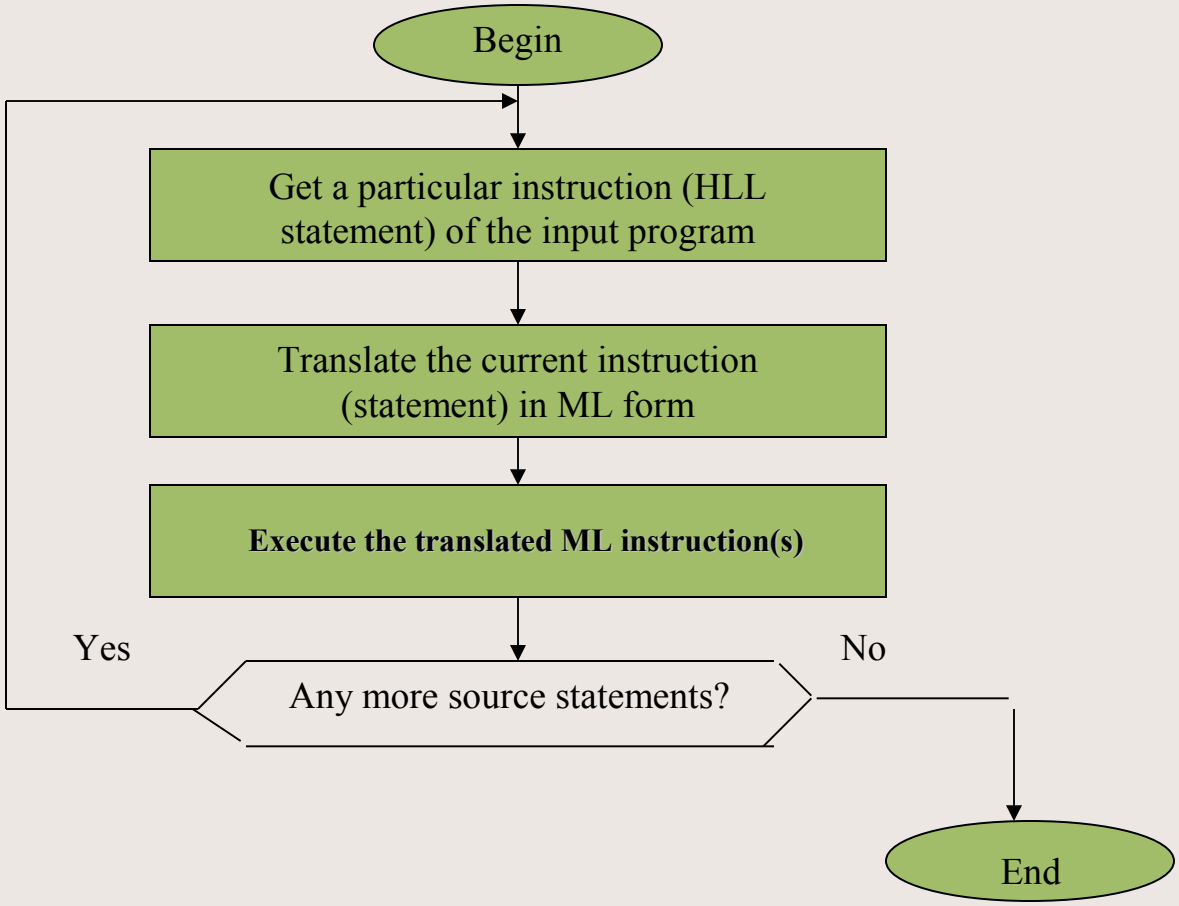
Обобщен алгоритъм на компилатор



Обобщен алгоритъм на интерпретатор

1. Начало.
2. Четене на EBH инструкция (HLL statement) от вх. Програма.
3. Превод на текущата инструкция в ME формат.
4. **Директно изпълнение на превода.**
5. Има ли още инструкции на входа? Ако Да, премини към 2. Иначе, премини към 6.
6. Край.

Обобщен алгоритъм на интерпретатор



Сравнение К. - И.

- И. се пише по-лесно от К.
- Изпълнението на една компилирана програма е по-бързо от изпълнението на една интерпретирана програма.
- К: Фаза компилация и изпълнение са отделни.
- И: Фаза компилация и изпълнение са съвместени.
- К: Веднъж компилирана, програма може многократно да се изпълнява.
- И: Изпълнение винаги се съпътства с превод (транслация).

Преносимост на ПО

- Идея и определение
- Езикът UNCOL (УНКОЛ)
- Компилация от полу интерпретиращ тип
- Практически примери
  - Pascal, P-code
  - MBPL, MK
  - Java, byte code
  - .NET, MSIL

Преносимост на софтуера

Една програма е преносима, ако може относително лесно да се пренесе от един компютър на друг компютър.

Т.е. усилията за пренасяне са много по-малко от усилията за пре-написването ѝ.

Преносимостта на софтуера е мярка за усилията, с които една програма се пренася от една изчислителна среда в друга различна изчислителна среда.

Преносимост на софтуера

Една програма е адаптируема, ако има процедура за нейната настройка в съответствие със специфични изисквания на потребителя.

Технически аспекти на адаптируемостта на софтуера са различните *Setup/Install* процедури за инсталиране на програмни продукти.

Проблемът UNCOL

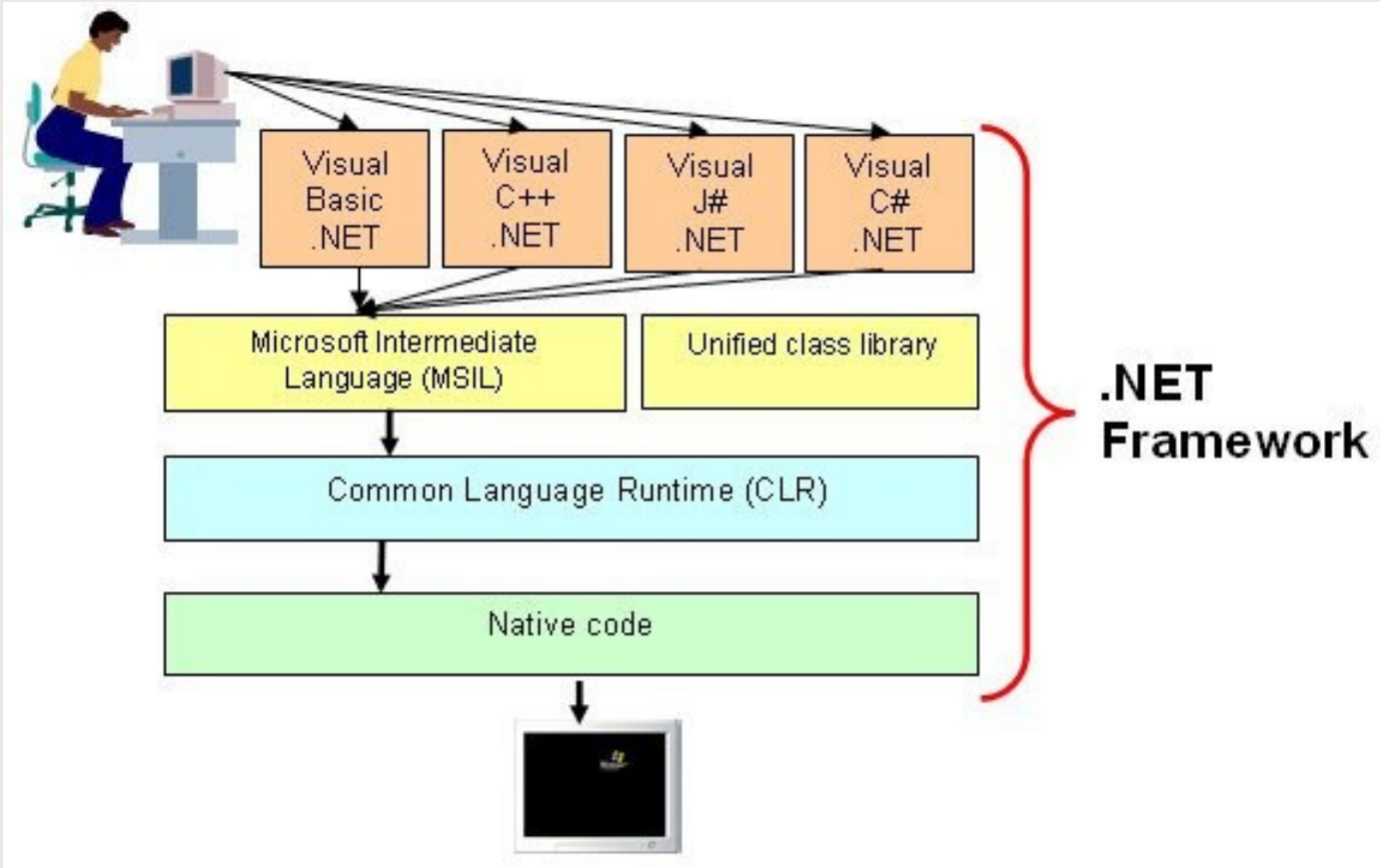
Проблемът UNCOL: реализацията на N*M езикови процесори да се редуцира до N+M езикови процесори.

Решението: да се въведе език от междинно ниво **UNCOL** (**UN**iversal **C**omputer **O**riented **L**anguage).

Езици от междинно ниво

Идеята за междинно ниво резултира в следната хибридна схема – компилация от полу- интерпретиращ тип

Приложения с междинен код – примери на PASCAL, MBPL, JAVA, Microsoft .NET


Visual Basic .NET
Visual C++ .NET
Visual J# .NET
Visual C# .NET
Microsoft Intermediate Language (MSIL)
Unified class library
Common Language Runtime (CLR)
Native code
.NET Framework

Т - Диаграми

- Т-диаграми за описание на ЕП (К. и И.)

Source language,
Target language,
Implementation language.

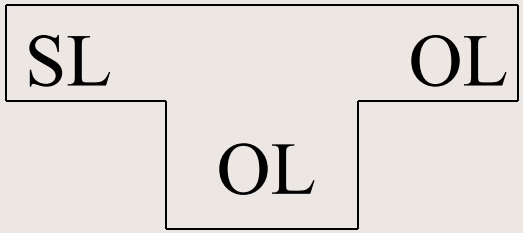
Резидентен компилатор;

Крос компилатор;

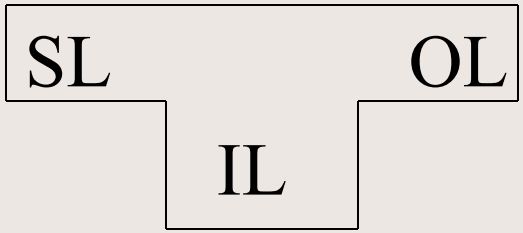
Само компилируем (Self compiling or bootstrapped) компилатор

Т - Диаграми

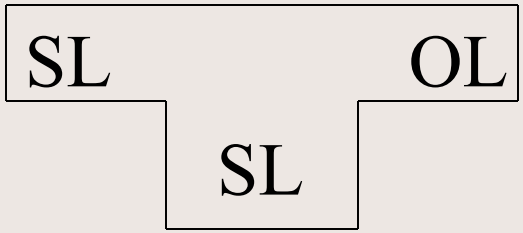
Резидентен Компилатор



Крос Компилатор



Само компилируем К.



Част 2

Въведение в Теория на Формалните Езици

Теория на множествата

1. Концепцията множество.

Определение: Множеството е съвкупност от обекти (атоми).

Атомите са членове (принадлежат) на множеството.

Теория на множествата

1. Задаване на множество:
   - Като изброим списък от елементи
     - A = { a, b, c, d, e }
   - Чрез предикат
     - B = { n | n е просто число }
   - Контекстно указване на липсващи елементи
     - C = { a, aa, aaa, aaaa, … }

Теория на множествата

1. Елементи на множество – атоми и/или множества
2. Подмножества /Subsets/ и надмножества /Supersets/
3. Равни (равномощни) множества
4. Две специални множества:
   - Универсално множество E
   - Празно множество (null set)

Теория на множествата

1. Дадено множество A.
2. Въвежда се: множество на всички подмножества на A.
3. Нарича се power set of A

   $p(A) = 2^A$
4. Мощността на множеството p(A)

   $pow(p(A)) = pow(2^A) = 2^{pow(A)}$

Теория на множествата

1. Дадено множество A = { a, b, c }
2. Илюстрация на pow(2^A) = $2^{pow(A)} = 2^3 = 8$
3. Съставяме всички възможни подмножества:

a	b	c		
0	0	0	{ }	– празно множество
0	0	1	{ c }	– подмножество /subset of A/
0	1	0	{ b }	– подмножество /subset of A/
0	1	1	{ b, c }	– подмножество /subset of A/
1	0	0	{ a }	– подмножество /subset of A/
1	0	1	{ a, c }	– подмножество /subset of A/
1	1	0	{ a, b }	– подмножество /subset of A/
1	1	1	{ a, b, c }	– самото множество /set A/

Теория на множествата

1. Операции над множества.
   - Сечение (пресичащи се, непресичащи се м-ва)
   - Обединение
   - Разлика B-A, относително допълнение на B в A
   - Допълнение на A до универсалното множество E, т.е. Разликата E-A се нарича абсолютно допълнение на A.

Теория на множествата

Наредена двойка – съдържа два обекта в определен ред. Описва се като (x,y)

Равенство на две наредени двойки (x,y) = (u,v)

Декартово произведение на две множества (Cartesian product) е множество от наредени двойки

Cartesian product of A and B, i.e. AxB

A={1, 2} B={3,4} AxB={ (1,3), (1,4), (2,3), 2,4) }

Теория на множествата

Теория на множествата

- S е крайно множество, ако е равномощно с множеството {1, 2, 3, … n }.
- S е безкрайно множество, ако е равномощно с някое свое собствено подмножество.
- S е изброимо множество, ако е равномощно с множеството на ест. числа {1, 2, 3, …}.
- Безкрайно множество, което не е изброимо м-во, е не изброимо множество.

Въведение в ТФЕ

- Азбука
- Низ
- Език

Въведение в ТФЕ

- Азбука
  - Определение
  - Означение
  - Мощност на азбука
  - Примери

Азбука

- Определение
  - Азбука е непразно крайно множество елементи, наречени символи.
- Означение
  - Големи букви от гръцката азбука Σ, Δ, Ώ
- Мощност на азбука
  - Σ = { a, b, c, d }
  - pow(Σ)=4 или #Σ=4

Азбука

- Примери
  - Binary alphabet B = { 0, 1 }
  - Octal
  - Decimal
  - Hexadecimal
  - Latin
  - Cyrillic
  - Abstract

Въведение в ТФЕ

- Низ
  - Определение
  - Означение
  - Дължина на низ
  - Операции над низове
  - Примери

Низ

- Определение
  - Всяка крайна последователност от символи, принадлежащи на дадена азбука, се нарича низ.
- Примери: дадена азбука $\Sigma = \{ a, b, c , d \}$
  - Правилни низове: a
  - Правилни низове: ca
  - Правилни низове: ccba
  - Правилни низове: bbb

Низ

- Конвенция:
- малки букви от началото на латиницата *a*, *b*, *c*, ... означават символи на азбуката;
- малки букви от опашката на латиницата *t*, *u*, *v*, *w*, ... означват символни низове
- големи латински букви *A*, *B*, *C*,… означават множества от низове:
- Примери:
  - u = a
  - v = ca
  - w = ccba
  - z = bbb

Низ

- Дължина на низ

Азбуки се характеризират с мощност.

Низове се характеризират с дължина.

u = a	|u| = 1
v = ca	|v| = 2
w = ccba	|w| = 4
z = bbb	|z| = 3

Низ

- Празен низ;

Низ, който не съдържа символи, без съдържание

Означение: e или ε

Празният низ е с нулева дължина, |e|=|ε|=0

- Низ, съставен от един и същ символ, повторен n пъти, може да се означи съкратено u = aaa…a = a^n
- Низ, съдържащ *n* символа *a*, се означава a^n ($a^1 = a$, $a^2 = aa$, $a^0 = \varepsilon$).

Низ

- Операции над низове:
  - Слепване (concatenation, catenation) ;
  - Обръщане (reversal).

Низ

- Операции над низове:
  - Слепване (concatenation, catenation) ;
    - Ако са дадени низовете x и y, тяхната конкатенация се бележи с $x.y$ или само xy. Получава се като препишем символите на втория низ след символите на първия. Частен случай е $x\varepsilon = \varepsilon x = x$, където x е произволен низ.
    - Низът $xy = acab$ е слепване на низовете $x = ac$ и $y = ab$.

Низ

- Операции над низове:
  - Обръщане (reversal).
    - Ако е зададен низ $x = a_1a_2\ldots a_n$, тогава *обърнатият низ* на низа x се бележи с x^R и е равен на $x^R = a_n\ldots a_2a_1$.
    - Валидно е $\varepsilon^R = \varepsilon$.

Низ

- Поднизове
  - Нека *x*, *y* и *z* са произволни низове в азбуката Σ
  - *x* се нарича *префикс* (или глава) на низа *xy*.
  - *y* се нарича *суфикс* (или опашка) на низа *xy*.
  - Низът *y* се нарича *подниз* на низа *xyz*.
  - Префиксът и суфиксът също са поднизове на низа.
  - Празният низ е суфикс, префикс и подниз на кой да е низ.

Забележка

Термините *знак* и *буква* се използват понякога като синоними на термина *символ*.

Термините *редица* и *дума* се използват понякога като синоними на термина *низ*.

Забележка

Низът не е множество (отделни символи се повтарят в низа).

Интуитивно въвеждане на специално множество
Нека е дадена азбука Σ.

Σ^* означава множеството на всички низове, вкл празния низ, съставени със символи на азбуката Σ.

Въведение в ТФЕ

- Език
  - Определение
  - Означение
  - Крайни и безкрайни езици
  - Универсалното множество от всички символни низове в една азбука
  - Примери

Формален език

- Определение. Един формален език L представлява множество от низове в азбука Σ.
- Това множество може да е крайно или безкрайно, но във всеки случай един език L е подмножество на Σ*.
- Всяка първична програма е последователност от низове, но не всяка последователност от низове е синтактично вярна програма.

Формален език

Множество L от низове с крайна дължина над дадена азбука Σ се нарича *език*. Теоретично това множество може да е крайно или безкрайно, но при всички случаи $L \subseteq \Sigma^*$, т.е. един език L представлява подмножество на множеството Σ^*.

Формален език

- Операции над формални езици
- ФЕ са множества. Следователно в сила са известните операции над множества:
  - Обединение
  - Сечение
  - Разлика
  - Допълнение
  - Подмножество (subset), надмножество (superset)

Въведение в ТФЕ

- Език
  - Начини за описание на формални езици
    - Изброяване на съставящите низове
    - Граматика (синтезираща система)
    - Разпознавател (анализираща система)

Описание на език според A.Aho & J.Ullman

Един ФЕ е определен , ако са известни:

- Азбука – множество от символи;
- Синтаксис – множество от синтактично верни програми;
- Семантика – смисъл (значение) на всички синтактично верни програми.

Операции над множества от низове

Наред със споменатите операции над множества (обединение, сечение, разлика, допълнение), се въвеждат три спецфични операции над множества от символни низове.

Нека означим като операнди P и Q две множества от низове.

- Дизюнкция (Alternation);
- Конкатенация (Concatenation);
- Итерация (Kleene closure or full iteration);
- Непразна итерация (Positive closure or non-empty iteration);

Операции над множества от символни низове

- Дизюнкция
- Конкатенация
- Итерация

Операции над множества от символни низове

- Дизюнкция (бинарна операция с операнди P и Q)
  - Определение

$$Z = P \mid Q$$

  - Примери

Операции над множества от символни низове

- Конкатенация (бинарна операция с операнди P и Q)
  - Определение

$$Z = P \,.\, Q = P\,Q$$

  - Примери

P={0,1} Q={a,b} PQ={0a, 0b, 1a, 1b}

Операции над множества от символни низове

- Итерация (унарна операция с операнд P)
  - Определение

$$Z = P^* = \{\varepsilon\} \mid P \mid P.P \mid P.P.P \mid \ldots$$

  - Примери

Операции над множества от символни низове

- Непразна итерация (унарна операция с операнд P)
  - Определение

$$Z = P^{+} = P \mid P.P \mid P.P.P \mid \ldots$$

  - Примери

Операции над множества от символни низове

- Релация: итерация – непразна итерация

$$P^* = \{\varepsilon\} \mid P \mid P.P \mid P.P.P \mid \ldots$$

$$P^+ = P \mid P.P \mid P.P.P \mid \ldots$$

$$P^+ = P \,.\, P^*$$

Операции над множества от символни низове

- Приоритет на операциите:

Висок

↑ Итерация - Iteration

↑ Конкатенация - Concatenation

↑ Дизюнкция - Alternation

Нисък

Примери за ФЕ като множества от низове:

- Зададени чрез характеристично свойство:

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

$L_1 = \{ \varepsilon, a, b, ab, abc \}$

$L_2 = \{ \varepsilon \}$

$L_3 = \{ a^n |_{n>=0} \} = \{ \varepsilon, a, aa, aaa, aaaa, \ldots \}$

$L_4 = \{ a^n |_{n>=1} \} = \{ a, aa, aaa, aaaa, \ldots \}$

$L_5 = \{a^n b^n |_{n >= 1} \} =$

$L_6 = \{a^n b^m |_{n, m >= 1} \} =$

$L_7 = \{ a^n |_{n \text{ е просто число}} \} =$

Примери за ФЕ като множества от низове:

- Зададени чрез операции над м-ва от низове:

$L_{11} = \{ a \}^*$

$L_{12} = \{ a \}^* . \{ b \} = \{ a \}^* \{ b \}$

$L_{13} = \{ a \} . \{ b \}^* = \{ a \} \{ b \}^*$

$L_{14} = \{ a \}^* . \{ b \}^* = \{ a \}^* \{ b \}^*$

$L_{15} = \{ b \}^+$

$L_{16} = \{ a \} . \{ b \}^+ . \{ c \} = \{ a \} \{ b \}^+ \{ c \}$

$L_{17} = \{ a, b \}^*$

$L_{18} = \{ b, a \}^*$

Благодаря
За
Вниманието