1.

1. Semantic analysis
2. Preprocessor
3. Lexical analysis
4. Left most derivation
5. Lexical analysis
6. Symbol table
7. زیرا برای parse کردن بعضی قواعد نیازمندیم تا جلوتر آن‌ها را ببینیم تا بتوانیم تصمیم بگیریم
8. یعنی تعدادی پایانه و ناپایانه را با استفاده از یک قاعده به ناپایانه معادل تبدیل کنیم

2.

1. اگر یک گره داشته باشد

*هر دو به 3 بخش پذیرند*

*فرض می‌کنیم برای* n *گره درست باشد و* E *دارای* K *گره باشد*

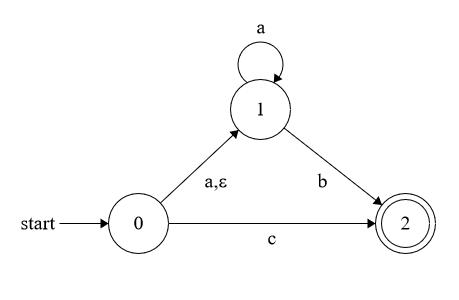
برود تا F دارای k+1 گره باشد

همانطور که قابل ملاضه است F هم به سه بخش پذیر است

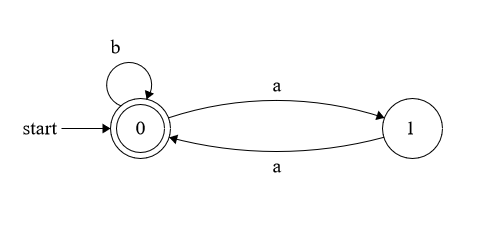
1. به جز عدد 0 تمام رشته‌های دودویی قابل تقسیم به 3 را درست می‌کند

3.









1. خیر – این زبان دارای شکلی به صورت پس باید تعداد a های اول و آخر به هم وابسته هستند پس نمی‌تواند عبارت منظم باشد

4.

1. با استفاده از عبارت منظم می‌توان NFA ساخت پس عبارت منظم را برعکس می‌کنیم تنها مشکلی که دارد این است که عبارت منظم می‌تواند به چند state مقصد برسد برای این که معکوس آن را در NFA درست کنیم یک state جدید می‌گذاریم و به تمام state های مقصد عبارت منظم اصلی با یال وصل می‌کنیم

و نمونه

از چپ به راست:

از راست به چپ:

5.

1. زیرا این گرامر دارای left recursion است
2. برای این کار گرامر نباید left recursion داشته باشد

void S**()** **{**

int isave **=** inputPointer**;**

**if** **(**inputSymbol **==** 'a'**)** **{**

advance**();**

**return** **true;**

**}**

inputPointer **=** isave**;**

**if** **(**inputSymbol **==** '('**)** **{**

advance**();**

**if** **(**A**())** **{**

**if** **(**inputSymbol **==** ')'**)** **{**

advance**();**

**return** **true;**

**}** **else** **{**

inputPointer **=** isave**;**

**return** FALSE**;**

**}**

**}** **else** **{**

**if** **(**inputSymbol **==** ')'**)** **{**

advance**();**

**return** **true;**

**}** **else** **{**

inputPointer **=** isave**;**

**return** **false;**

**}**

**}**

**}**

**}**

6.

1. ابتدا immediate left recursion سپس left factoring انجام می‌دهیم

7.

1. مقادیر اعداد زیر را به ترتیب چاپ می‌کند

**{**

t1 **=** newtemp**()**

t2 **=** newtemp**()**

t1**.**place **=** initial**.**place **–** 1

t2**.**place **=** final**.**place

S.nextlist = M3.quad

backpatch**(**S1**.**nextlist**,**M1**.**quad**)**

emit**(**'if' t1**.**place '>=' t2**.**place 'goto' S**.**nextlist**)**

emit**(**t1**.**place '=' t1**.**place '+' 1**)**

emit**(**'goto' M2**.**quad**)**

**}**

8.

هنوز نحوه ساختن کد سه آدرسه از روش تجزیهی بازگشتی نزولی تدریس نشده است