Parsing Expression Grammars:

An Example Driven Introduction

By

Mackenzie High

Spring 2013

Part 1 of 2: Initial Questions

What is a Grammar?

A grammar is a pattern that describes strings.

What is a parse-tree?

A parse-tree is a tree that maps rules in a grammar to substrings in a string.

What does a parser do?

Given a grammar and an input string, a parser creates a parse-tree.

Let's See an Example!

Example Grammar

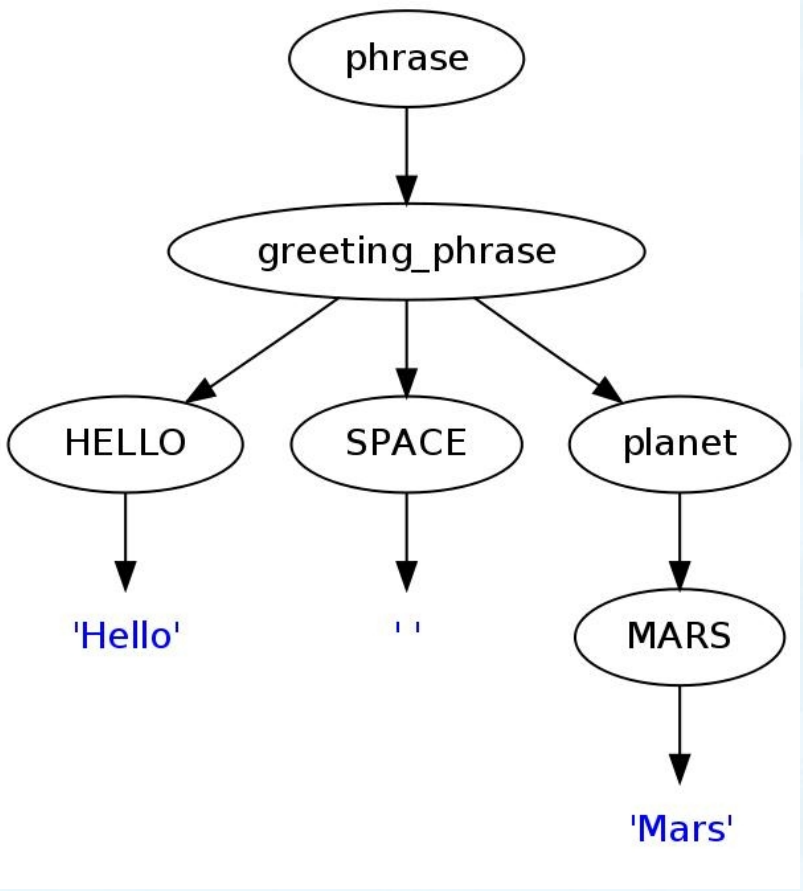
```
phrase = greeting_phrase / parting_phrase;
greeting_phrase = HELLO , SPACE , planet;
parting_phrase = GOODBYE , SPACE , planet;
planet = MARS / VULCAN;
HELLO = "Hello";
GOODBYE = "Bye Bye";
SPACE = ' ';
MARS = "Mars";
VULCAN = "Vulcan";
```

Example #1

Input

Hello Mars

Parse Tree

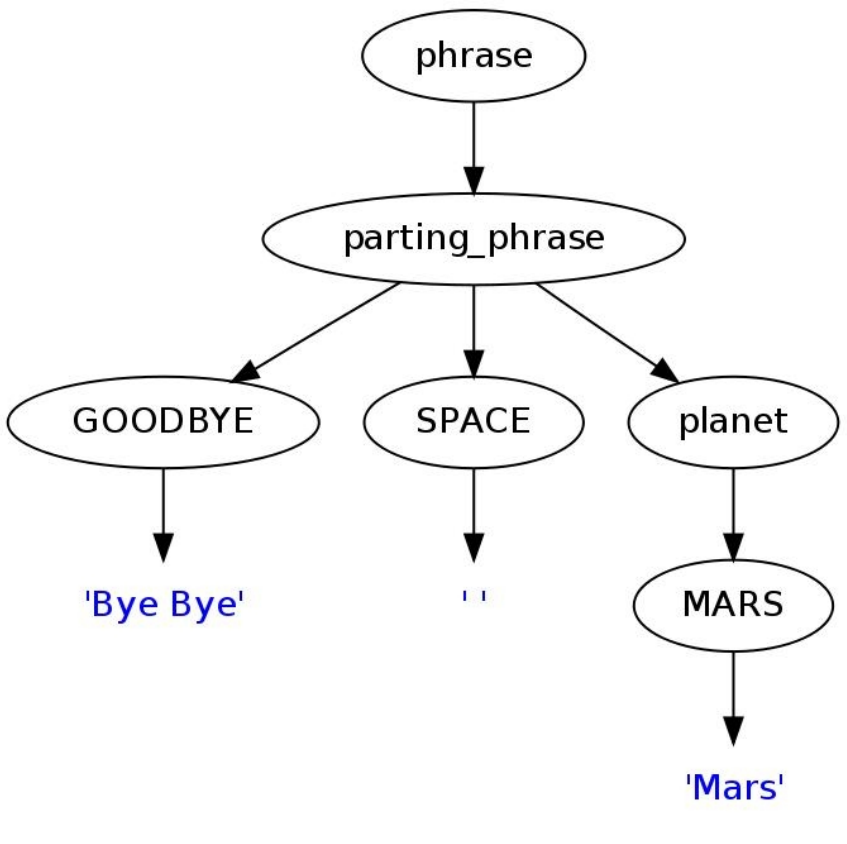


Example #2

Input

Bye Bye Mars

Parse Tree

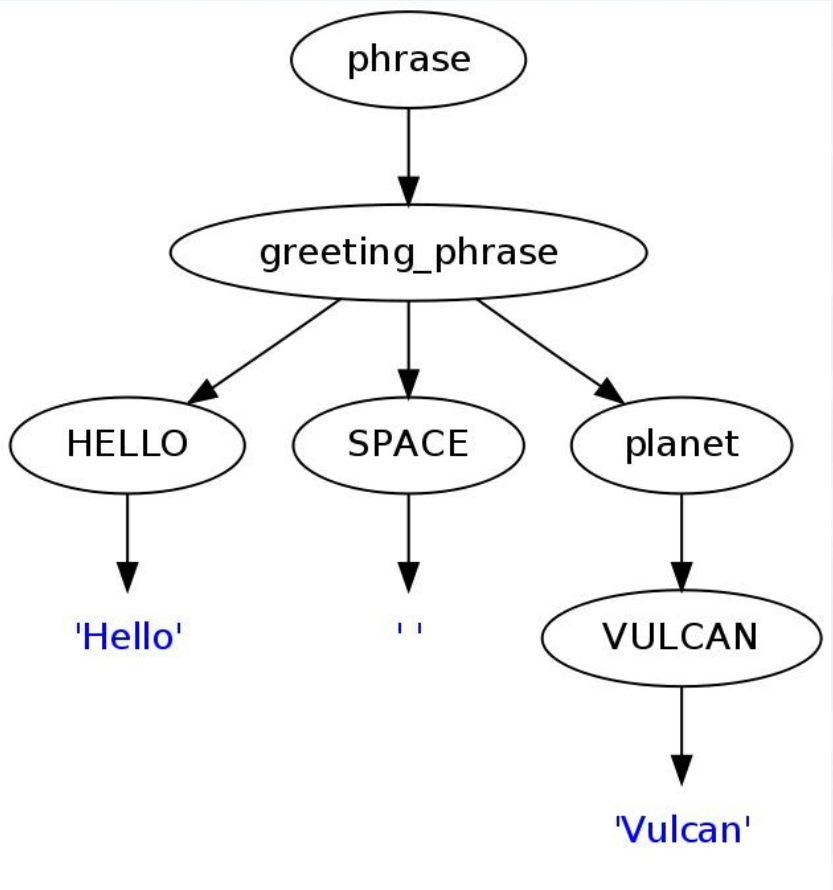


Example #3

Input

Hello Vulcan

Parse Tree

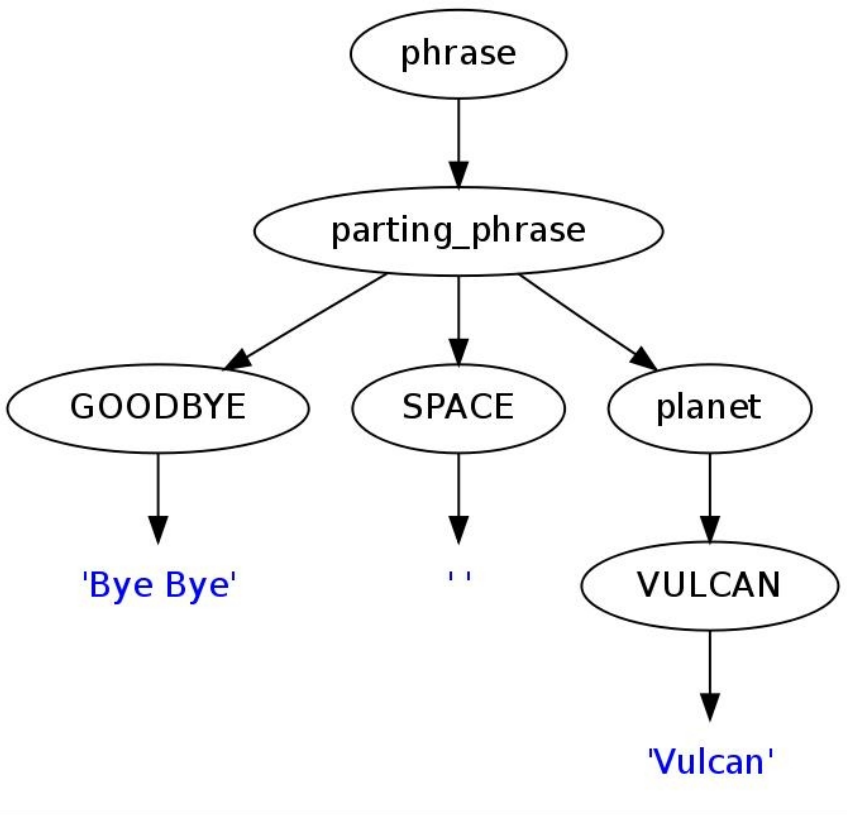


Example #4

Input

Bye Bye Vulcan

Parse Tree



Example #5

Input

Goodbye Mars

Parse Tree

No parse tree exists, because the grammar does not describe this input.

In other words, a parse-error occurred.

Part 2 of 2: Grammars in Detail

Character Rules

Syntax Form #1

name = '*character*';

(where *character* is literally a character)

Syntax Form #2

name = '*minimum*' – '*maximum*';

Where:

1. *minimum* is literally a character.
2. *maximum* is literally a character.
3. *minimum* <= *maximum*

Description

Form #1 defines a grammar rule that can match a single predefined character.

Form #2 defines a grammar rule that can match an inclusive range of of characters.

Example Grammar

S = A , B , C;

A = 'X';

B = '0' - '9';

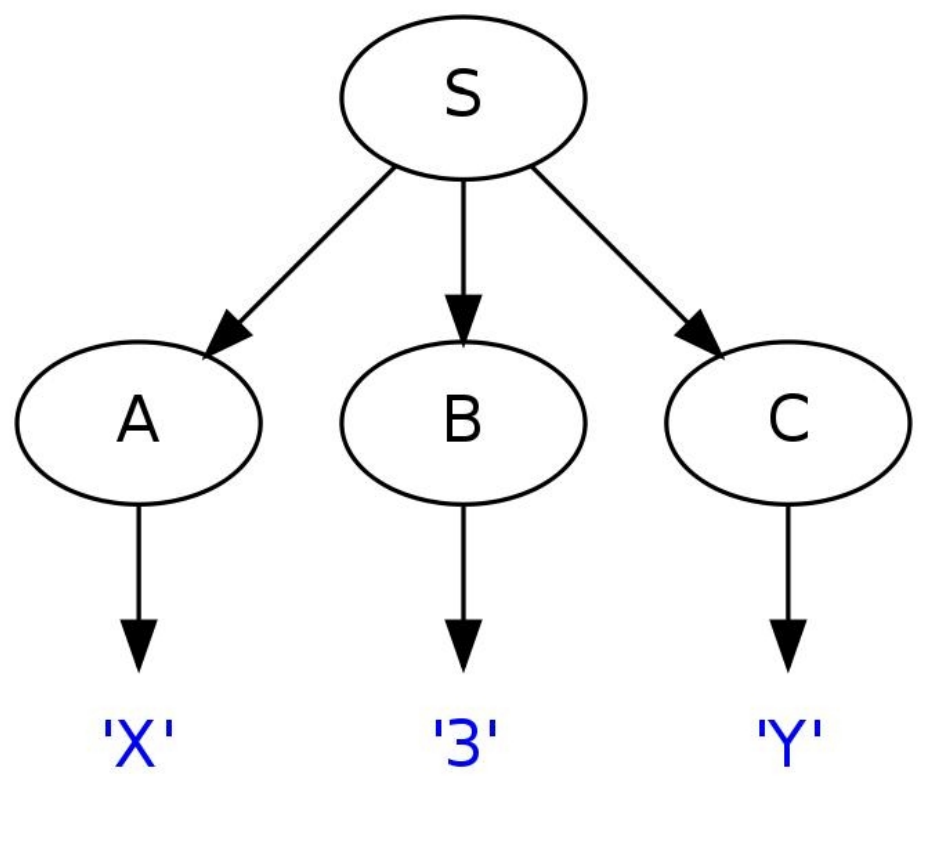
C = 'Y';

Example #1

Input

X3Y

Parse Tree

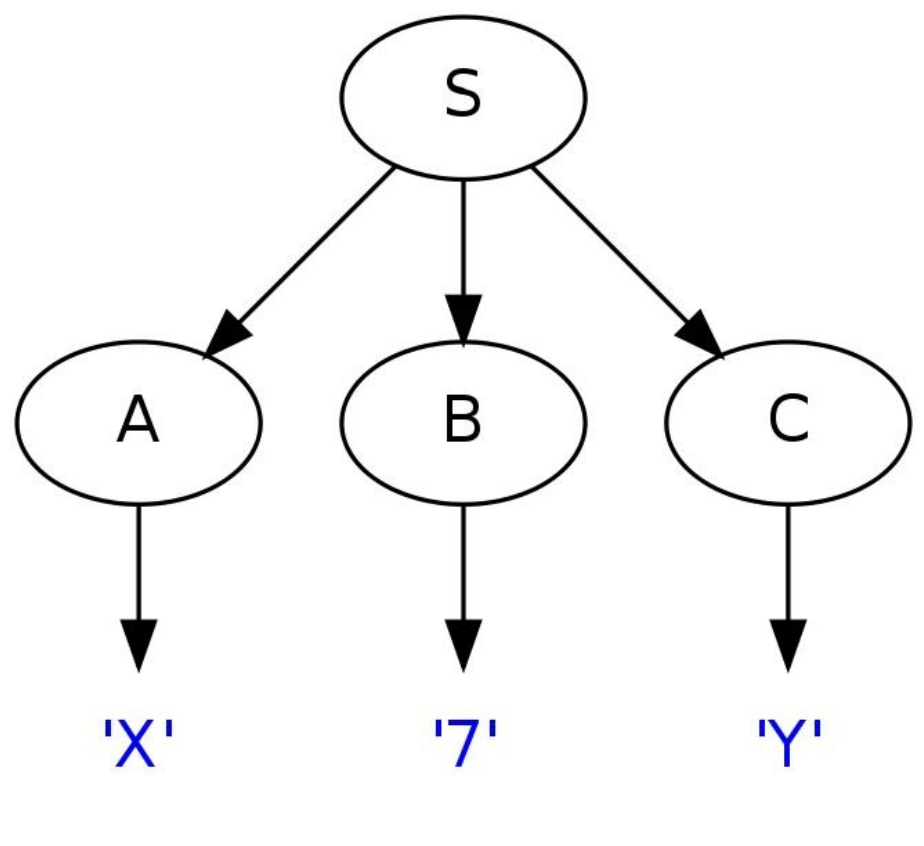


Example #2

Input

X7Y

Parse Tree



String Rules

Syntax

name = “*A string of characters*”;

(The string can be empty.)

Description

As the name implies, a string-rule defines a grammar rule that can match a predefined series of characters.

Example Grammar

word = prefix , suffix;

prefix = “afore”;

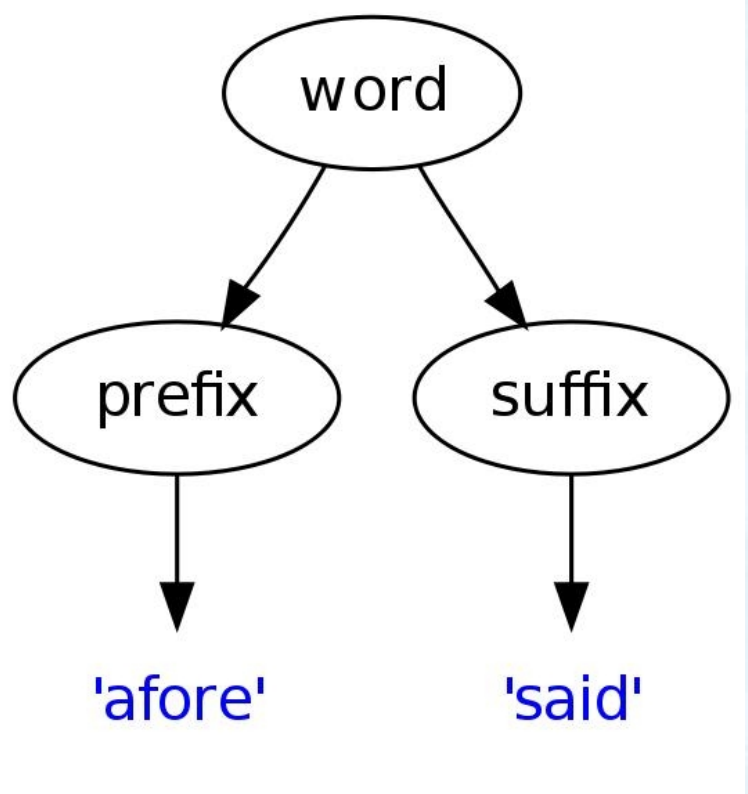
suffix = “said”;

Example

Input

aforesaid

Parse Tree



Sequence Rules

Syntax

name = *element1* , *element2, …* , *elementN*;

Where:

1. N >= 1

2. *elementX* is the name of another rule.

Description

A sequence-rule defines a grammar rule that matches a sequence of other grammar rules.

Example Grammar

M = T , T;

T = P , S;

P = 'A' – 'Z';

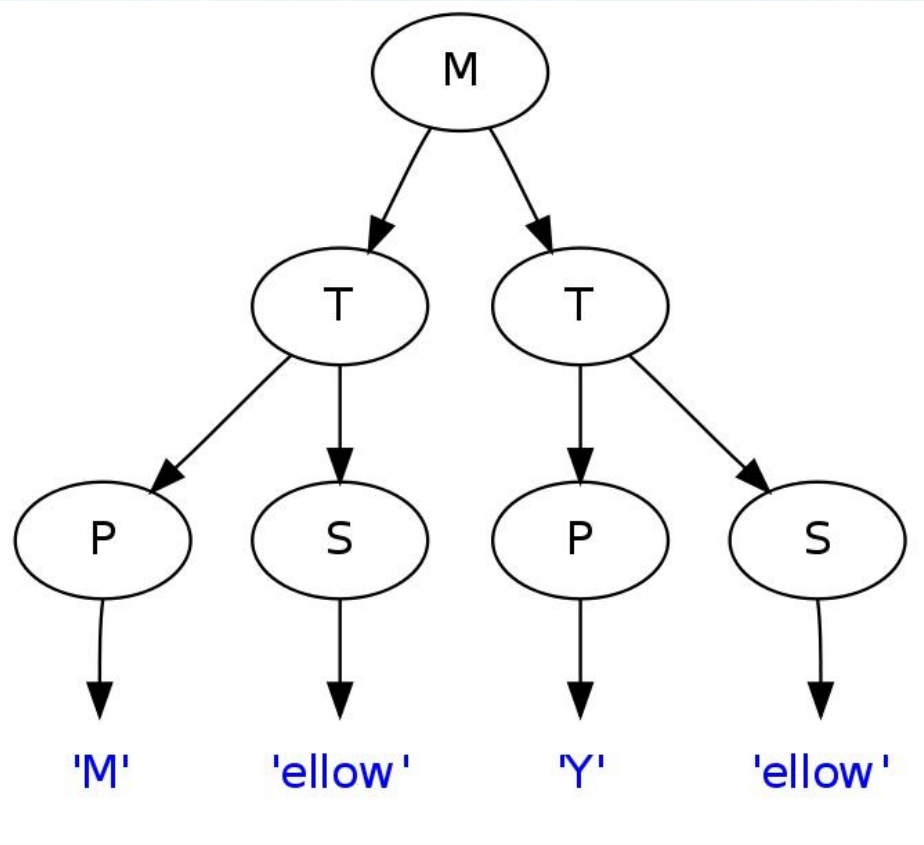
S = “ellow”;

Example

Input

MellowYellow

Parse Tree



Choice Rules

Syntax

name = *option1* / *option2* / … / *optionN*;

Where:

1. N >= 2

2. *optionX* is the name of another rule.

Description

A choice-rule attempts to match each of its options in the order they are written, until one of them succeeds. At which time, the choice-rule itself succeeds. If none of the options succeed, then the choice-rule itself fails.

Order is Important

Parsing Expression Grammars are unique in that choice-rules attempt to match the options in a predefined order.

Order is Important

Other types of grammars, such as EBNF, do not define the order in which the options will be checked.

Order is Important

Be sure to remember this fact, when examining grammars that are not Parsing Expression Grammars.

Differing Syntax

In other forms of grammars, such as EBNF, choice-rules use a slightly different syntax:

name = *option1* | *option2* | ... | *optionN*;

Example Grammar

C = G , D;

D = W / E;

G = “Go”;

W = “West”;

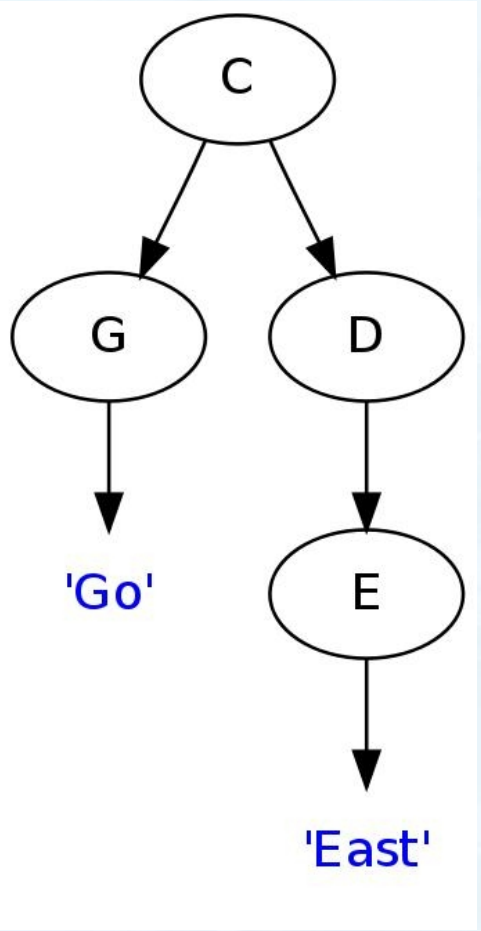
E = “East”;

Example #1

Input

GoEast

Parse Tree

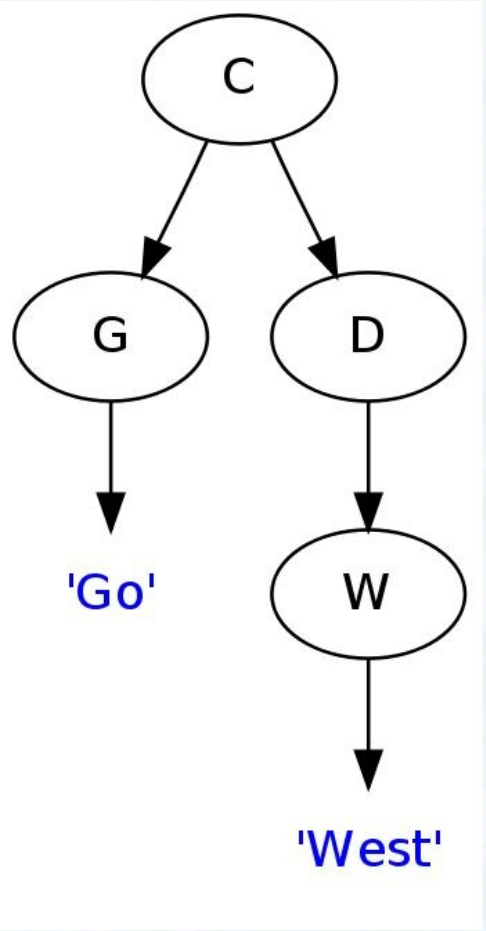


Example #2

Input

GoWest

Parse Tree



Repetition Rules

Syntax Form #1 (Bounded-Repetition-Rules)

name = *item* { *minimum* , *maximum* };

Where:

1. *item* is the name of another rule.
2. *minimum* is an unsigned integer.
3. *maximum* is an unsigned integer.
4. *minimum* < *maximum*

Syntax Form #2 (Option-Rules)

name = *item* ?;

Equivalence:

name = *item* { 0 , 1 }

Syntax Form #2
(Zero-OR-More Rules)

name = *item* *;

Equivalence:

name = *item* { 0 , ∞ }

Syntax Form #2
(One-OR-More Rules)

name = *item* +;

Equivalence:

name = *item* { 1 , ∞ }

Description

A repetition-rule matches, if and only if, another specified grammar rule matches at least a minimum number of times up to a maximum number of times.

Note

The zero-or-more and one-or-more forms are arguably the most important repetition rules, because they provide a concise method of defining infinite sequences.

Example Grammar #1

hex = prefix , digits;

digits = digit { 1 , 4 };

prefix = “0x”;

digit = N / L;

N = '0' – '9';

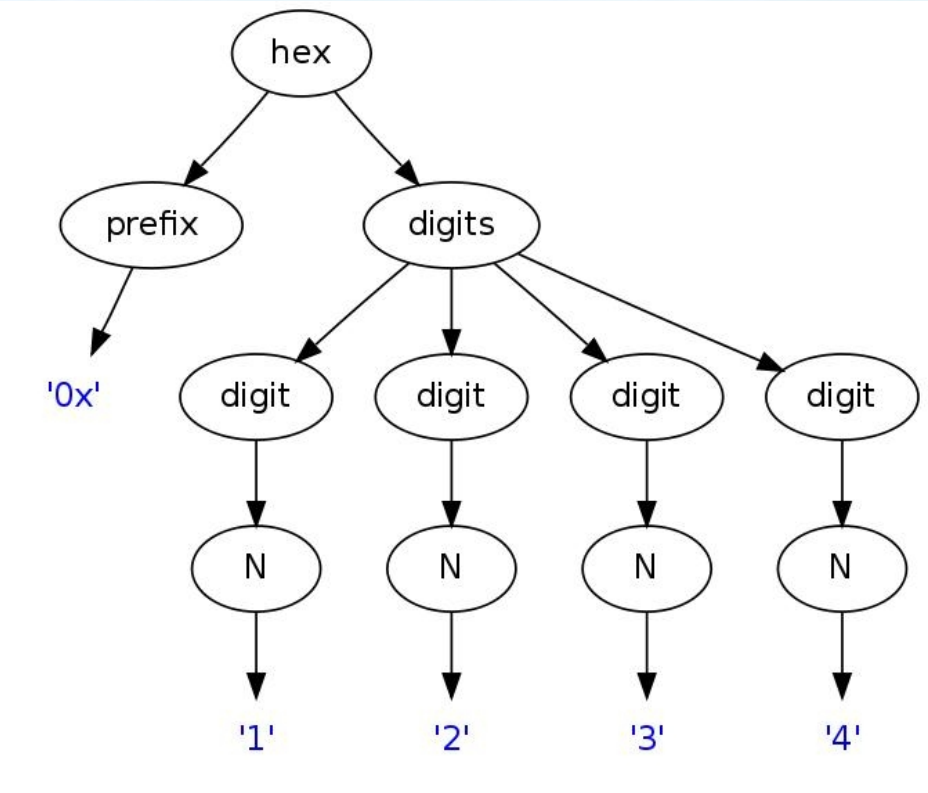
L = 'A' - 'F';

Example #1

Input

0x1234

Parse Tree

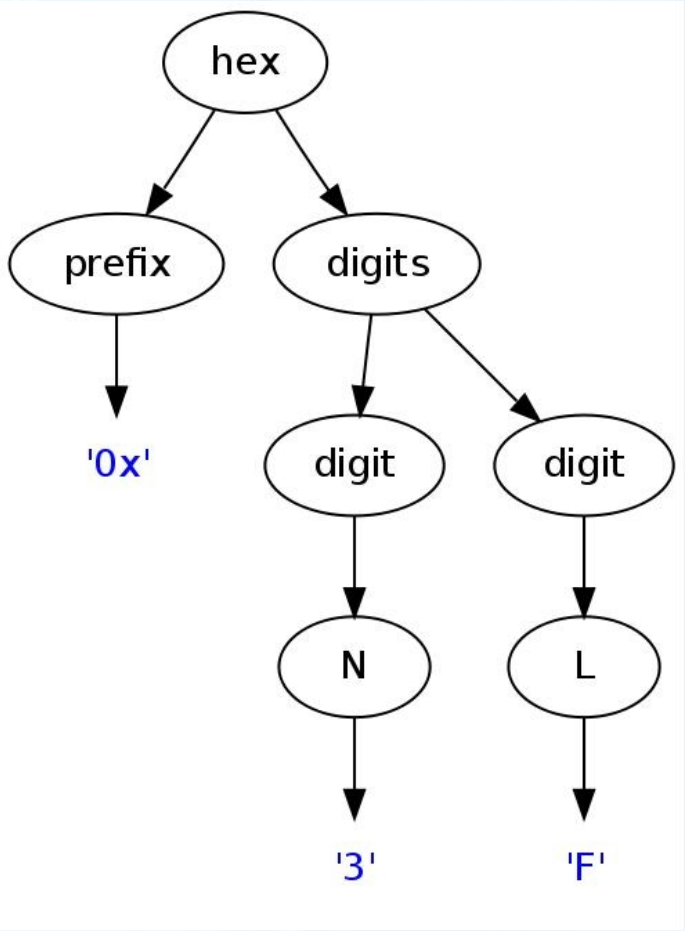


Example #2

Input

0x3F

Parse Tree



Example Grammar #2

name = first , middle , last;

first = “Jorge”;

middle = T ?;

T = “Mario”;

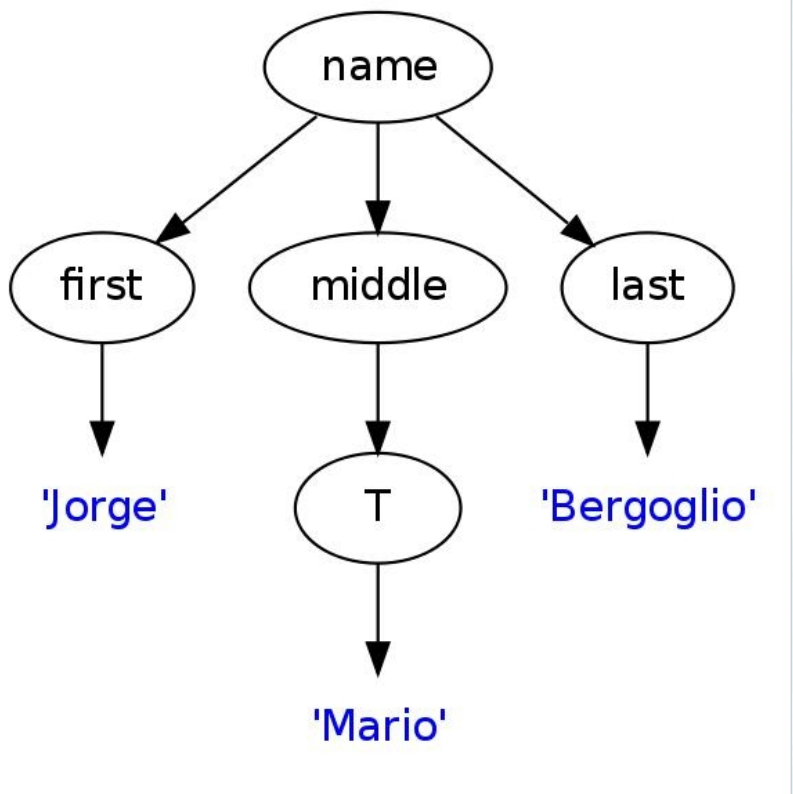
last = “Bergoglio”;

Example #1

Input

JorgeMarioBergoglio

Parse Tree

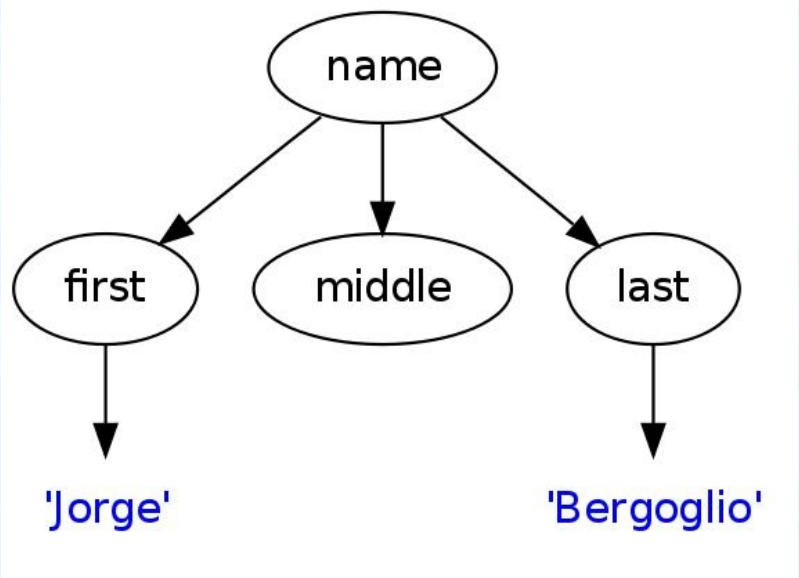


Example #2

Input

JorgeBergoglio

Parse Tree



The End!