Context-Free Grammars

Grammars were invented by Chomsky in the mid-1950s for describing natural languages. In the late 1950s, one of Chomsky's types of grammar (Type 2 or ***context-free*** grammar) was reinvented by Backus and proposed as a way to specify the syntax of the new language Algol.

Context-Free Grammars

Grammars were invented by Chomsky in the mid-1950s for describing natural languages. In the late 1950s, one of Chomsky's types of grammar (Type 2 or ***context-free*** grammar) was reinvented by Backus and proposed as a way to specify the syntax of the new language Algol.

This proposal was adopted in the **Algol 60 Report** (edited by Naur), an influential document considered to have done an excellent job of specifying Algol 60.

Context-Free Grammars

Grammars were invented by Chomsky in the mid-1950s for describing natural languages. In the late 1950s, one of Chomsky's types of grammar (Type 2 or ***context-free*** grammar) was reinvented by Backus and proposed as a way to specify the syntax of the new language Algol.

This proposal was adopted in the **Algol 60 Report** (edited by Naur), an influential document considered to have done an excellent job of specifying Algol 60.

The grammar notation used in the Algol 60 Report is now called "Backus Naur Form" or **BNF**.

Context-Free Grammars

Grammars were invented by Chomsky in the mid-1950s for describing natural languages. In the late 1950s, one of Chomsky's types of grammar (Type 2 or ***<u>context-free</u>*** grammar) was reinvented by Backus and proposed as a way to specify the syntax of the new language Algol.

This proposal was adopted in the **Algol 60 Report** (edited by Naur), an influential document considered to have done an excellent job of specifying Algol 60.

The grammar notation used in the Algol 60 Report is now called "Backus Naur Form" or **BNF**.

Like many authors (but unlike Sethi), we use the term ***BNF*** more loosely, to simply mean "*a commonly used notation for writing context-free grammars*"; we refer to grammars written in such a notation as ***BNF specifications***.

A grammar written in BNF notation on p. 46 of Sethi (p. 47 in the course reader).

On p. 42, Sethi gives this equivalent grammar that is written in a similar notation. ***<u>We will consider this notation to be BNF</u>***,

$$
\begin{aligned}
\langle expression \rangle \quad ::= \quad & \langle expression \rangle + \langle term \rangle \\
| \quad & \langle expression \rangle - \langle term \rangle \\
| \quad & \langle term \rangle \\
\langle term \rangle \quad ::= \quad & \langle term \rangle * \langle factor \rangle \\
| \quad & \langle term \rangle \,/\, \langle factor \rangle \\
| \quad & \langle factor \rangle \\
\langle factor \rangle \quad ::= \quad & \mathbf{number} \\
| \quad & \mathbf{name} \\
| \quad & (\ \langle expression \rangle\ )
\end{aligned}
$$

A grammar written in BNF notation on p. 46 of Sethi (p. 47 in the course reader).

Figure 2.10 BNF syntactic rules for arithmetic expressions.

On p. 42, Sethi gives this equivalent grammar that is written in a similar notation. ***We will consider this notation to be BNF***,

$$
\begin{aligned}
E \quad ::= \quad & E + T \mid E - T \mid T \\
T \quad ::= \quad & T * F \mid T \,/\, F \mid F \\
F \quad ::= \quad & \mathbf{number} \mid \mathbf{name} \mid (\ E\ )
\end{aligned}
$$

Figure 2.6 A grammar for arithmetic expressions.

$$
\begin{aligned}
\langle expression \rangle \ &::= \ \langle expression \rangle + \langle term \rangle \\
&\ \ \mid \ \ \langle expression \rangle - \langle term \rangle \\
&\ \ \mid \ \ \langle term \rangle \\
\langle term \rangle \ &::= \ \langle term \rangle * \langle factor \rangle \\
&\ \ \mid \ \ \langle term \rangle \, / \, \langle factor \rangle \\
&\ \ \mid \ \ \langle factor \rangle \\
\langle factor \rangle \ &::= \ \mathbf{number} \\
&\ \ \mid \ \ \mathbf{name} \\
&\ \ \mid \ \ (\ \langle expression \rangle \ )
\end{aligned}
$$

A grammar written in BNF notation on p. 46 of Sethi (p. 47 in the course reader).

Figure 2.10 BNF syntactic rules for arithmetic expressions.

On p. 42, Sethi gives this equivalent grammar that is written in a similar notation. ***We will consider this notation to be BNF***, even though it isn't exactly the same as the notation used in the Algol 60 Report and so Sethi does ***not*** call it BNF.

$$
\begin{aligned}
E \ &::= \ E + T \ \mid \ E - T \ \mid \ T \\
T \ &::= \ T * F \ \mid \ T \, / \, F \ \mid \ F \\
F \ &::= \ \mathbf{number} \ \mid \ \mathbf{name} \ \mid \ (\ E \ )
\end{aligned}
$$

Figure 2.6 A grammar for arithmetic expressions.

We will use the term ***<u>grammar</u>*** to mean "context-free grammar"; we will not use other types of grammar.

-
-
-
-

We will use the term ***grammar*** to mean "context-free grammar"; we will not use other types of grammar.

- A grammar is a relatively concise way to precisely specify certain (possibly infinite) *sets of finite sequences of symbols*; those symbols are referred to as ***terminals*** of the grammar.
-
-
-

We will use the term ***grammar*** to mean "context-free grammar"; we will not use other types of grammar.

- A grammar is a relatively concise way to precisely specify certain (possibly infinite) *sets of finite sequences of symbols*; those symbols are referred to as ***terminals*** of the grammar.
- Each of the specified *sets of finite sequences of terminals* is denoted by a ***nonterminal*** of the grammar.
-
-

We will use the term ***grammar*** to mean "context-free grammar"; we will not use other types of grammar.

- A grammar is a relatively concise way to precisely specify certain (possibly infinite) *sets of finite sequences of symbols*; those symbols are referred to as ***terminals*** of the grammar.
- Each of the specified *sets of finite sequences of terminals* is denoted by a ***nonterminal*** of the grammar.
- One of the nonterminals is regarded as the “most important”: It is called the ***starting nonterminal*** (or *start symbol* or *sentence symbol*);
-

We will use the term ***grammar*** to mean "context-free grammar"; we will not use other types of grammar.

- A grammar is a relatively concise way to precisely specify certain (possibly infinite) *sets of finite sequences of symbols*; those symbols are referred to as ***terminals*** of the grammar.
- Each of the specified *sets of finite sequences of terminals* is denoted by a ***nonterminal*** of the grammar.
- One of the nonterminals is regarded as the “most important”: It is called the ***starting nonterminal*** (or *start symbol* or *sentence symbol*); the set of sequences of terminals it denotes is called the ***language*** *generated by* (or *language of*) the grammar.
-

We will use the term ***grammar*** to mean "context-free grammar"; we will not use other types of grammar.

- A grammar is a relatively concise way to precisely specify certain (possibly infinite) *sets of finite sequences of symbols*; those symbols are referred to as ***terminals*** of the grammar.
- Each of the specified *sets of finite sequences of terminals* is denoted by a ***nonterminal*** of the grammar.
- One of the nonterminals is regarded as the "most important": It is called the ***starting nonterminal*** (or *start symbol* or *sentence symbol*); the set of sequences of terminals it denotes is called the ***language*** *generated by* (or *language of*) the grammar.
- We commonly think of the other nonterminals as auxiliary nonterminals that are defined for use in defining the starting nonterminal.

$$
\begin{aligned}
\langle \textit{real-number} \rangle &::= \langle \textit{integer-part} \rangle \,.\, \langle \textit{fraction} \rangle \\
\langle \textit{integer-part} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{integer-part} \rangle \langle \textit{digit} \rangle \\
\langle \textit{fraction} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{digit} \rangle \langle \textit{fraction} \rangle \\
\langle \textit{digit} \rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

In the above grammar:

$$\langle \textit{real-number} \rangle ::= \langle \textit{integer-part} \rangle \;.\; \langle \textit{fraction} \rangle$$
$$\langle \textit{integer-part} \rangle ::= \langle \textit{digit} \rangle \mid \langle \textit{integer-part} \rangle \langle \textit{digit} \rangle$$
$$\langle \textit{fraction} \rangle ::= \langle \textit{digit} \rangle \mid \langle \textit{digit} \rangle \langle \textit{fraction} \rangle$$
$$\langle \textit{digit} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

In the above grammar:

The following characters are the 11 ***terminals***:

. 0 1 2 3 4 5 6 7 8 9

A ***terminal*** of a grammar is a constant symbol that is ***not*** defined by the grammar.

⟨*real-number*⟩ ::= ⟨*integer-part*⟩ . ⟨*fraction*⟩
⟨*integer-part*⟩ ::= ⟨*digit*⟩ | ⟨*integer-part*⟩ ⟨*digit*⟩
⟨*fraction*⟩ ::= ⟨*digit*⟩ | ⟨*digit*⟩ ⟨*fraction*⟩
⟨*digit*⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

In the above grammar:

The following characters are the 11 ***terminals***:

. 0 1 2 3 4 5 6 7 8 9

A ***terminal*** of a grammar is a constant symbol that is ***not*** defined by the grammar.

The following are the 4 ***nonterminals***:

<real-number> *<integer-part>* *<fraction>* *<digit>*

A ***nonterminal*** of a grammar is a variable that denotes a *set of finite sequences of terminals*. For example, *<digit>* denotes

⟨*real-number*⟩ ::= ⟨*integer-part*⟩ . ⟨*fraction*⟩
⟨*integer-part*⟩ ::= ⟨*digit*⟩ | ⟨*integer-part*⟩ ⟨*digit*⟩
⟨*fraction*⟩ ::= ⟨*digit*⟩ | ⟨*digit*⟩ ⟨*fraction*⟩
⟨*digit*⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

In the above grammar:

The following characters are the 11 ***terminals***:

. 0 1 2 3 4 5 6 7 8 9

A ***terminal*** of a grammar is a constant symbol that is ***not*** defined by the grammar.

The following are the 4 ***nonterminals***:

<real-number> *<integer-part>* *<fraction>* *<digit>*

A ***nonterminal*** of a grammar is a variable that denotes a *set of finite sequences of terminals*. For example, *<digit>* denotes the set {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}.

$$
\begin{array}{rcl}
\langle \textit{real-number} \rangle & ::= & \langle \textit{integer-part} \rangle \;.\; \langle \textit{fraction} \rangle \\
\langle \textit{integer-part} \rangle & ::= & \langle \textit{digit} \rangle \mid \langle \textit{integer-part} \rangle \, \langle \textit{digit} \rangle \\
\langle \textit{fraction} \rangle & ::= & \langle \textit{digit} \rangle \mid \langle \textit{digit} \rangle \, \langle \textit{fraction} \rangle \\
\langle \textit{digit} \rangle & ::= & 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{array}
$$

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

In the above grammar:

$$
\begin{aligned}
\langle \textit{real-number} \rangle &::= \langle \textit{integer-part} \rangle \, . \, \langle \textit{fraction} \rangle \\
\langle \textit{integer-part} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{integer-part} \rangle \langle \textit{digit} \rangle \\
\langle \textit{fraction} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{digit} \rangle \langle \textit{fraction} \rangle \\
\langle \textit{digit} \rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

In the above grammar:

There are 15 rules called ***productions***. Each production:

- has a left side that is a *single nonterminal*, and
- has a right side that is a *sequence of 0 or more terminals and/or nonterminals*.

The “vertical bar” symbol | means:

⟨*real-number*⟩ ::= ⟨*integer-part*⟩ . ⟨*fraction*⟩
⟨*integer-part*⟩ ::= ⟨*digit*⟩ | ⟨*integer-part*⟩ ⟨*digit*⟩
⟨*fraction*⟩ ::= ⟨*digit*⟩ | ⟨*digit*⟩ ⟨*fraction*⟩
⟨*digit*⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

In the above grammar:

There are 15 rules called ***productions***. Each production:

- has a left side that is a *single nonterminal*, and
- has a right side that is a *sequence of 0 or more terminals and/or nonterminals*.

The “vertical bar” symbol | means:

The left side of this production is the same as the left side of the ***previous*** *production.*

Example: The **3rd** production of the above grammar is

⟨*real-number*⟩ ::= ⟨*integer-part*⟩ . ⟨*fraction*⟩
⟨*integer-part*⟩ ::= ⟨*digit*⟩ | ⟨*integer-part*⟩ ⟨*digit*⟩
⟨*fraction*⟩ ::= ⟨*digit*⟩ | ⟨*digit*⟩ ⟨*fraction*⟩
⟨*digit*⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

In the above grammar:

There are 15 rules called ***productions***. Each production:

- has a left side that is a *single nonterminal*, and
- has a right side that is a *sequence of 0 or more terminals and/or nonterminals*.

The “vertical bar” symbol | means:

*The left side of this production is the same as the left side of the **previous** production.*

Example: The **3rd** production of the above grammar is

<integer-part> ::= *<integer-part> <digit>*

$$
\begin{aligned}
\langle \textit{real-number} \rangle &::= \langle \textit{integer-part} \rangle \,.\, \langle \textit{fraction} \rangle \\
\langle \textit{integer-part} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{integer-part} \rangle \langle \textit{digit} \rangle \\
\langle \textit{fraction} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{digit} \rangle \langle \textit{fraction} \rangle \\
\langle \textit{digit} \rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

⟨*real-number*⟩	::=	⟨*integer-part*⟩ . ⟨*fraction*⟩
⟨*integer-part*⟩	::=	⟨*digit*⟩ | ⟨*integer-part*⟩ ⟨*digit*⟩
⟨*fraction*⟩	::=	⟨*digit*⟩ | ⟨*digit*⟩ ⟨*fraction*⟩
⟨*digit*⟩	::=	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

Grammar notation is "free format": We can insert whitespace characters, ***including newlines***, between symbols without changing the specified grammar!

For example, the 2nd and 3rd productions

<integer-part> ::= *<digit>* | *<integer-part> <digit>*

of the above grammar could be ***<u>rewritten</u>*** as:

⟨*real-number*⟩ ::= ⟨*integer-part*⟩ . ⟨*fraction*⟩
⟨*integer-part*⟩ ::= ⟨*digit*⟩ | ⟨*integer-part*⟩ ⟨*digit*⟩
⟨*fraction*⟩ ::= ⟨*digit*⟩ | ⟨*digit*⟩ ⟨*fraction*⟩
⟨*digit*⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

Grammar notation is "free format": We can insert whitespace characters, ***including newlines***, between symbols without changing the specified grammar!

For example, the 2nd and 3rd productions

<integer-part> ::= <digit> | <integer-part> <digit>

of the above grammar could be ***rewritten*** as:

<integer-part> ::= <digit>
| <integer-part> <digit>

⟨real-number⟩ ::= *⟨integer-part⟩* . *⟨fraction⟩*
⟨integer-part⟩ ::= *⟨digit⟩* | *⟨integer-part⟩* *⟨digit⟩*
⟨fraction⟩ ::= *⟨digit⟩* | *⟨digit⟩* *⟨fraction⟩*
⟨digit⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

Grammar notation is “free format”: We can insert whitespace characters, ***including newlines***, between symbols without changing the specified grammar!

For example, the 2nd and 3rd productions

<integer-part> ::= <digit> | <integer-part> <digit>

of the above grammar could be ***rewritten*** as:

<integer-part> ::= <digit>
| <integer-part> <digit>

Intuitively, a production *N* ::= ... means “any ... is an *N*”.

$$
\begin{aligned}
\langle \textit{real-number} \rangle &::= \langle \textit{integer-part} \rangle \; . \; \langle \textit{fraction} \rangle \\
\langle \textit{integer-part} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{integer-part} \rangle \langle \textit{digit} \rangle \\
\langle \textit{fraction} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{digit} \rangle \langle \textit{fraction} \rangle \\
\langle \textit{digit} \rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

$$
\begin{aligned}
\langle \textit{real-number} \rangle &::= \langle \textit{integer-part} \rangle \; . \; \langle \textit{fraction} \rangle \\
\langle \textit{integer-part} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{integer-part} \rangle \langle \textit{digit} \rangle \\
\langle \textit{fraction} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{digit} \rangle \langle \textit{fraction} \rangle \\
\langle \textit{digit} \rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

<*real-number*> is the ***starting nonterminal*** of the above grammar.

In this course, we use the convention that ***unless otherwise indicated***, *the starting nonterminal of a grammar is …*

$$
\begin{aligned}
\langle \textit{real-number} \rangle &::= \langle \textit{integer-part} \rangle \;.\; \langle \textit{fraction} \rangle \\
\langle \textit{integer-part} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{integer-part} \rangle\, \langle \textit{digit} \rangle \\
\langle \textit{fraction} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{digit} \rangle\, \langle \textit{fraction} \rangle \\
\langle \textit{digit} \rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

<*real-number*> is the ***starting nonterminal*** of the above grammar.

In this course, we use the convention that ***unless otherwise indicated***, *the starting nonterminal of a grammar is the nonterminal on the left side of the **first** production*:

⟨*real-number*⟩ ::= ⟨*integer-part*⟩ . ⟨*fraction*⟩
⟨*integer-part*⟩ ::= ⟨*digit*⟩ | ⟨*integer-part*⟩ ⟨*digit*⟩
⟨*fraction*⟩ ::= ⟨*digit*⟩ | ⟨*digit*⟩ ⟨*fraction*⟩
⟨*digit*⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

<*real-number*> is the ***starting nonterminal*** of the above grammar.

In this course, we use the convention that ***unless otherwise indicated***, *the starting nonterminal of a grammar is the nonterminal on the left side of the* ***first*** *production*:

If you write a grammar and want *some other* nonterminal to be its starting nonterminal, then

$$
\begin{aligned}
\langle \textit{real-number} \rangle &::= \langle \textit{integer-part} \rangle \,.\, \langle \textit{fraction} \rangle \\
\langle \textit{integer-part} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{integer-part} \rangle \langle \textit{digit} \rangle \\
\langle \textit{fraction} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{digit} \rangle \langle \textit{fraction} \rangle \\
\langle \textit{digit} \rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

<real-number> is the ***starting nonterminal*** of the above grammar.

In this course, we use the convention that ***<u>unless otherwise indicated</u>***, *the starting nonterminal of a grammar is the nonterminal on the left side of the **<u>first</u>** production*:

If you write a grammar and want *some other* nonterminal to be its starting nonterminal, then you must ***<u>explicitly indicate</u>*** which nonterminal is the starting nonterminal!

$$
\begin{aligned}
\langle \textit{real-number} \rangle &::= \langle \textit{integer-part} \rangle \;.\; \langle \textit{fraction} \rangle \\
\langle \textit{integer-part} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{integer-part} \rangle \langle \textit{digit} \rangle \\
\langle \textit{fraction} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{digit} \rangle \langle \textit{fraction} \rangle \\
\langle \textit{digit} \rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

$$
\begin{aligned}
\langle \textit{real-number} \rangle &::= \langle \textit{integer-part} \rangle \; . \; \langle \textit{fraction} \rangle \\
\langle \textit{integer-part} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{integer-part} \rangle \langle \textit{digit} \rangle \\
\langle \textit{fraction} \rangle &::= \langle \textit{digit} \rangle \mid \langle \textit{digit} \rangle \langle \textit{fraction} \rangle \\
\langle \textit{digit} \rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

<*empty*> denotes the empty string; other people write ϵ or λ to denote the empty string.

Example:

⟨*real-number*⟩ ::= ⟨*integer-part*⟩ . ⟨*fraction*⟩
⟨*integer-part*⟩ ::= ⟨*digit*⟩ | ⟨*integer-part*⟩ ⟨*digit*⟩
⟨*fraction*⟩ ::= ⟨*digit*⟩ | ⟨*digit*⟩ ⟨*fraction*⟩
⟨*digit*⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Figure 2.3 BNF rules for real numbers.

A grammar given by Sethi to specify unsigned floating point literals in a simple language.

<*empty*> denotes the empty string; other people write ϵ or λ to denote the empty string.

Example: Changing the 2nd production above from *<integer-part>* ::= *<digit>* to *<integer-part>* ::= *<empty>* will allow a number with ***no digits before the point*** (e.g., .213) to belong to the language of the grammar.

Note that *<empty>* is ***neither*** a terminal ***nor*** a nonterminal!

Parse Trees

Q. Exactly which sequences of terminals belong to the set of sequences of terminals that is denoted by a given nonterminal ***N*** of a grammar?

Parse Trees

Q. Exactly which sequences of terminals belong to the set of sequences of terminals that is denoted by a given nonterminal ***N*** of a grammar?

A. A sequence of terminals t_1 ... t_k belongs to the set of sequences denoted by a nonterminal ***N*** *if and only if* …

Parse Trees

Q. Exactly which sequences of terminals belong to the set of sequences of terminals that is denoted by a given nonterminal ***N*** of a grammar?

A. A sequence of terminals t_1 ... t_k belongs to the set of sequences denoted by a nonterminal ***N*** *if and only if* there is a **parse tree with root *N* that generates** t_1 ... t_k.

Parse Trees

Q. Exactly which sequences of terminals belong to the set of sequences of terminals that is denoted by a given nonterminal ***N*** of a grammar?

A. A sequence of terminals $t_1 \dots t_k$ belongs to the set of sequences denoted by a nonterminal ***N*** *if and only if* there is a **parse tree with root *N* that generates** $t_1 \dots t_k$.

Unless otherwise indicated, the term ***parse tree*** means *parse tree <u>whose root is the starting nonterminal</u>.*

So we can say that a sequence of terminals $t_1 \dots t_k$ belongs to the language of a grammar *if and only if* there is a **parse tree that generates** $t_1 \dots t_k$.

Comment:

Parse Trees

Q. Exactly which sequences of terminals belong to the set of sequences of terminals that is denoted by a given nonterminal ***N*** of a grammar?

A. A sequence of terminals $t_1 \ldots t_k$ belongs to the set of sequences denoted by a nonterminal ***N*** *if and only if* there is a **parse tree with root *N* that generates** $t_1 \ldots t_k$.

Unless otherwise indicated, the term ***parse tree*** means *parse tree <u>whose root is the starting nonterminal</u>*.

So we can say that a sequence of terminals $t_1 \ldots t_k$ belongs to the language of a grammar *if and only if* there is a **parse tree that generates** $t_1 \ldots t_k$.

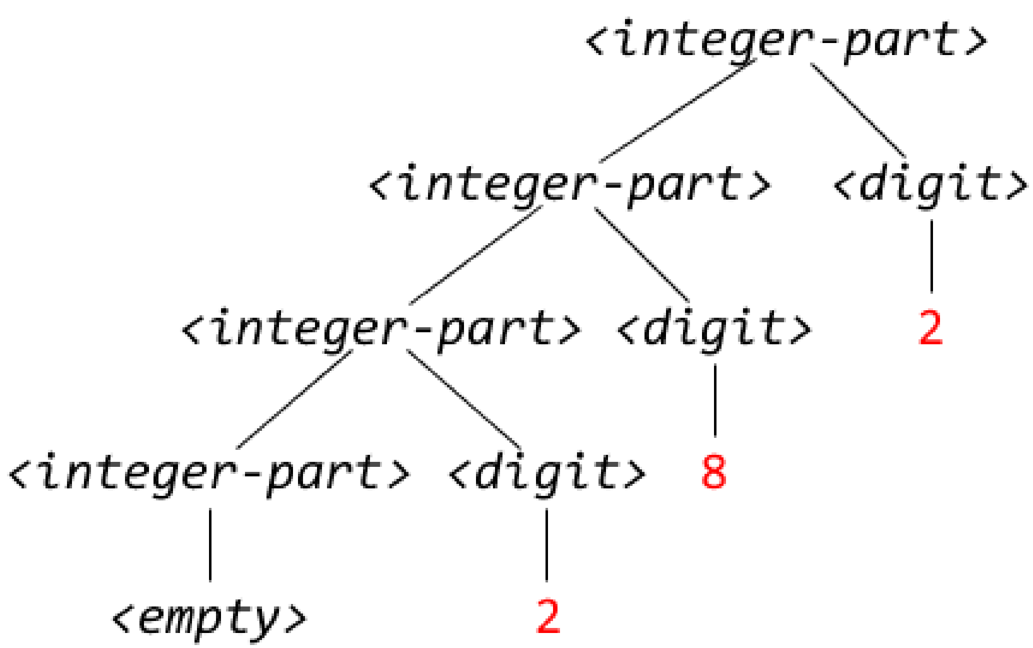
Comment: Instead of using parse trees, we can also answer the above question using the concept of a ***<u>derivation</u>*** that is introduced on pp. 40 – 41 of Sethi.

Below is a parse tree, whose root is <*integer-part*>, that shows 282 belongs to the set of sequences denoted by <*integer-part*> in the following grammar:

<*real-number*> ::= <*integer-part*> . <*fraction*>
<*integer-part*> ::= <*empty*> | <*integer-part*> <*digit*>
<*fraction*> ::= <*digit*> | <*digit*> <*fraction*>
<*digit*> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Below is a parse tree, whose root is *<integer-part>*, that shows 282 belongs to the set of sequences denoted by *<integer-part>* in the following grammar:

<real-number> ::= *<integer-part>* . *<fraction>*
<integer-part> ::= *<empty>* | *<integer-part>* *<digit>*
<fraction> ::= *<digit>* | *<digit>* *<fraction>*
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9



Note: This is just a picture to show what parse trees look like.

A precise definition of a parse tree will be given below.

Below is a parse tree that shows 282.83 belongs to the language of the same grammar:

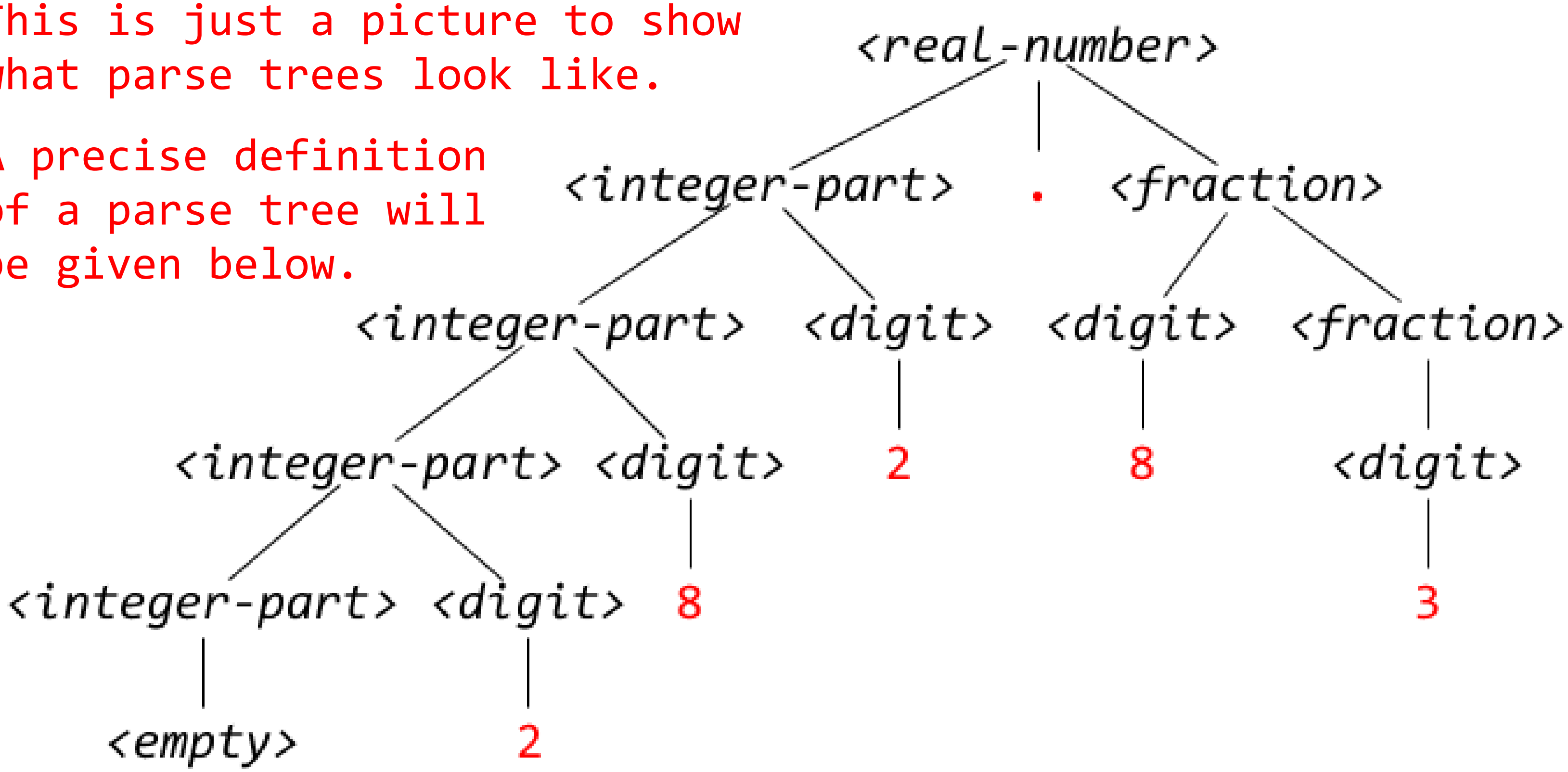
```
  <real-number> ::= <integer-part> . <fraction>
 <integer-part> ::= <empty> | <integer-part> <digit>
     <fraction> ::= <digit> | <digit> <fraction>
        <digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Below is a parse tree that shows 282.83 belongs to the language of the same grammar:

<real-number> ::= *<integer-part>* . *<fraction>*
<integer-part> ::= *<empty>* | *<integer-part>* *<digit>*
<fraction> ::= *<digit>* | *<digit>* *<fraction>*
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

This is just a picture to show what parse trees look like.

A precise definition of a parse tree will be given below.



Given a nonterminal ***N*** and a ***parse tree with root N*** is an ordered rooted tree with the following properties:

1.
2.

3.
4.

Given a nonterminal ***N*** and a ***parse tree with root N*** is an ordered rooted tree with the following properties:

1. The **root** is .
2. Each **leaf** either is or is ; moreover, .
3. Each **internal node** is .
4. The left-to-right sequence of children of any internal node ***M*** is .

Given a nonterminal ***N*** and a ***parse tree with root N*** is an ordered rooted tree with the following properties:

1. The **root** is the nonterminal ***N***.
2. Each **leaf** either is or is ; moreover, .
3. Each **internal node** is .
4. The left-to-right sequence of children of any internal node ***M*** is .

Given a nonterminal ***N*** and a ***parse tree with root N*** is an ordered rooted tree with the following properties:

1. The **root** is the nonterminal ***N***.
2. Each **leaf** either is a terminal or is *<empty>*; moreover, .
3. Each **internal node** is .
4. The left-to-right sequence of children of any internal node ***M*** is .

Given a nonterminal ***N*** and a <u>***parse tree with root N***</u> is an ordered rooted tree with the following properties:

1. The **root** is the nonterminal ***N***.
2. Each **leaf** either is a terminal or is *<empty>*; moreover, a leaf that is *<empty>* has no sibling.
3. Each **internal node** is .
4. The left-to-right sequence of children of any internal node ***M*** is

   .

Given a nonterminal ***N*** and a ***parse tree with root N*** is an ordered rooted tree with the following properties:

1. The **root** is the nonterminal ***N***.
2. Each **leaf** either is a terminal or is <*empty*>; moreover, a leaf that is <*empty*> has no sibling.
3. Each **internal node** is a nonterminal.
4. The left-to-right sequence of children of any internal node ***M*** is

   .

Given a nonterminal ***N*** and a ***parse tree with root N*** is an ordered rooted tree with the following properties:

1. The **root** is the nonterminal ***N***.
2. Each **leaf** either is a terminal or is <*empty*>; moreover, a leaf that is <*empty*> has no sibling.
3. Each **internal node** is a nonterminal.
4. The left-to-right sequence of children of any internal node ***M*** is the right side of a production whose left side is the nonterminal ***M***.

Given a nonterminal ***N*** and a ***parse tree with root N*** is an ordered rooted tree with the following properties:

1. The **root** is the nonterminal ***N***.
2. Each **leaf** either is a terminal or is <*empty*>; moreover, a leaf that is <*empty*> has no sibling.
3. Each **internal node** is a nonterminal.
4. The left-to-right sequence of children of any internal node ***M*** is the right side of a production whose left side is the nonterminal ***M***.

Unless otherwise indicated, the term ***parse tree*** means *parse tree whose root is the starting nonterminal*.

Given a nonterminal ***N*** and a ***parse tree with root N*** is an ordered rooted tree with the following properties:

1. The **root** is the nonterminal ***N***.
2. Each **leaf** either is a terminal or is <*empty*>; moreover, a leaf that is <*empty*> has no sibling.
3. Each **internal node** is a nonterminal.
4. The left-to-right sequence of children of any internal node ***M*** is the right side of a production whose left side is the nonterminal ***M***.

Unless otherwise indicated, the term ***parse tree*** means *parse tree whose root is the starting nonterminal*.

Given terminals t_1, ..., t_k, a ***parse tree with root N that generates*** t_1 ... t_k (or ***parse tree with root N of*** t_1 ... t_k) is a parse tree with root ***N*** for which …

Given a nonterminal N and a ***parse tree with root N*** is an ordered rooted tree with the following properties:

1. The **root** is the nonterminal N.
2. Each **leaf** either is a terminal or is <*empty*>; moreover, a leaf that is <*empty*> has no sibling.
3. Each **internal node** is a nonterminal.
4. The left-to-right sequence of children of any internal node M is the right side of a production whose left side is the nonterminal M.

Unless otherwise indicated, the term ***parse tree*** means *parse tree whose root is the starting nonterminal*.

Given terminals t_1, ..., t_k, a ***parse tree with root N that generates*** t_1 ... t_k (or ***parse tree with root N of*** t_1 ... t_k) is a parse tree with root N for which the left-to-right sequence of leaves that are not <*empty*> is t_1, ..., t_k.

RECALL:

Q. Exactly which sequences of terminals belong to the set of sequences of terminals that is denoted by a given nonterminal ***N*** of a grammar?

A.

RECALL:

Q. Exactly which sequences of terminals belong to the set of sequences of terminals that is denoted by a given nonterminal ***N*** of a grammar?

A. A sequence of terminals t_1 ... t_k belongs to the set of sequences denoted by a nonterminal ***N*** *if and only if* …

RECALL:

Q. Exactly which sequences of terminals belong to the set of sequences of terminals that is denoted by a given nonterminal ***N*** of a grammar?

A. A sequence of terminals $t_1 \ldots t_k$ belongs to the set of sequences denoted by a nonterminal ***N*** *if and only if* there is a **parse tree with root *N* that generates** $t_1 \ldots t_k$.

Let's draw a parse tree, whose root is <*integer-part*>, that shows 282 belongs to the set of sequences denoted by <*integer-part*> in the following grammar:

Let's draw a parse tree, whose root is *<integer-part>*, that shows 282 belongs to the set of sequences denoted by *<integer-part>* in the following grammar:

<real-number> ::= *<integer-part>* . *<fraction>*

<integer-part> ::= *<empty>* | *<integer-part>* *<digit>*

<fraction> ::= *<digit>* | *<digit>* *<fraction>*

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Let's draw a parse tree, whose root is *<integer-part>*, that shows 282 belongs to the set of sequences denoted by *<integer-part>* in the following grammar:

<real-number> ::= *<integer-part>* . *<fraction>*
<integer-part> ::= *<empty>* | *<integer-part>* *<digit>*
<fraction> ::= *<digit>* | *<digit>* *<fraction>*
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

The root of this parse tree is <integer-part>.

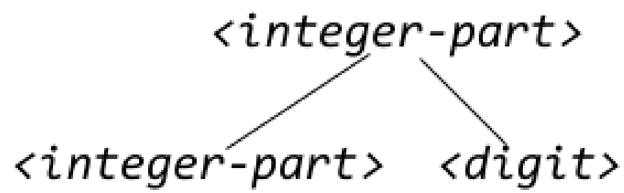
<integer-part>

Let's draw a parse tree, whose root is *<integer-part>*, that shows 282 belongs to the set of sequences denoted by *<integer-part>* in the following grammar:

<real-number> ::= *<integer-part>* . *<fraction>*
<integer-part>* ::=** *<empty>* | ***<integer-part> <digit>
<fraction> ::= *<digit>* | *<digit>* *<fraction>*
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

The left-to-right sequence of children of any internal node M is the right side of a production whose left side is the nonterminal M.

Using production:
<integer-part> ::= <integer-part> <digit>

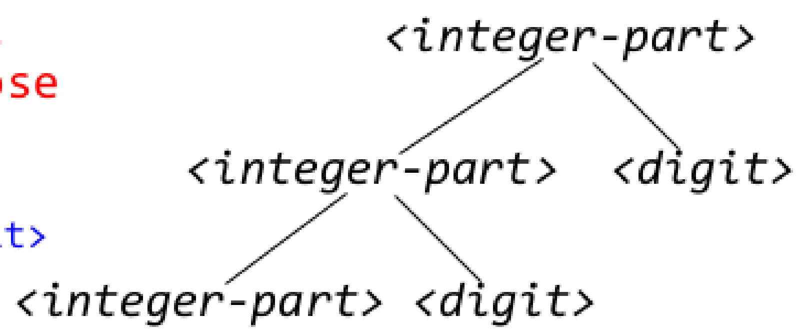


Let's draw a parse tree, whose root is *<integer-part>*, that shows 282 belongs to the set of sequences denoted by *<integer-part>* in the following grammar:

<real-number> ::= *<integer-part>* . *<fraction>*
<integer-part> ::= *<empty>* | ***<integer-part> <digit>***
<fraction> ::= *<digit>* | *<digit>* *<fraction>*
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

The left-to-right sequence of children of any internal node M is the right side of a production whose left side is the nonterminal M.

Re-using production:
<integer-part> ::= <integer-part> <digit>

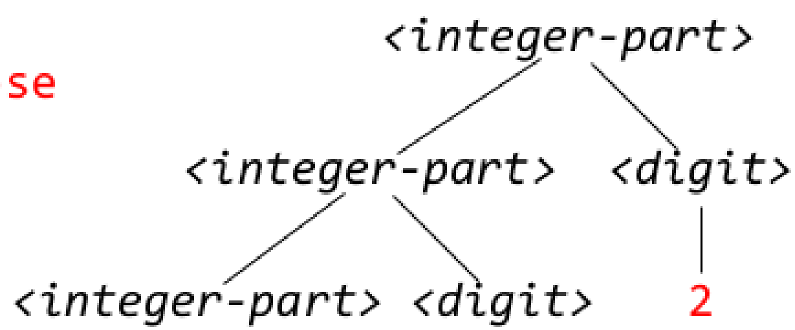


Let's draw a parse tree, whose root is *<integer-part>*, that shows 282 belongs to the set of sequences denoted by *<integer-part>* in the following grammar:

<real-number> ::= *<integer-part>* . *<fraction>*
<integer-part> ::= *<empty>* | *<integer-part>* *<digit>*
<fraction> ::= *<digit>* | *<digit>* *<fraction>*
<digit> **::=** 0 | 1 | **2** | 3 | 4 | 5 | 6 | 7 | 8 | 9

The left-to-right sequence of children of any internal node M is the right side of a production whose left side is the nonterminal M.

Using production:
<digit> ::= 2

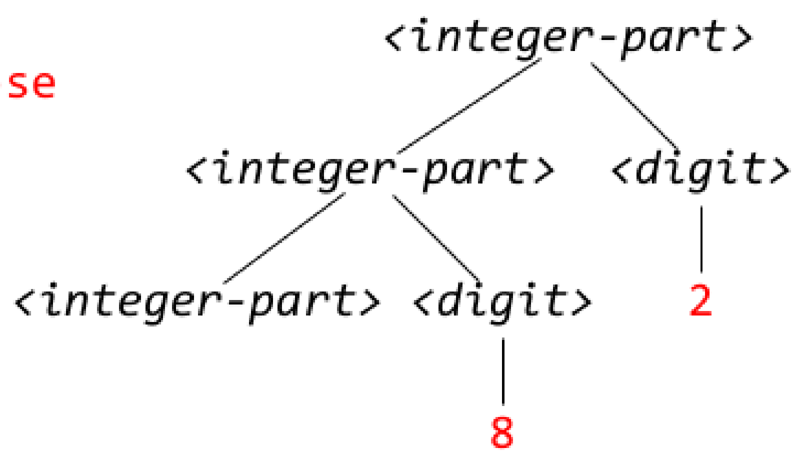


Let's draw a parse tree, whose root is *<integer-part>*, that shows 282 belongs to the set of sequences denoted by *<integer-part>* in the following grammar:

<real-number> ::= *<integer-part>* . *<fraction>*
<integer-part> ::= *<empty>* | *<integer-part>* *<digit>*
<fraction> ::= *<digit>* | *<digit>* *<fraction>*
***<digit>* ::=** 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | **8** | 9

The left-to-right sequence of children of any internal node M is the right side of a production whose left side is the nonterminal M.

Using production:
<digit> ::= 8



Let's draw a parse tree, whose root is *<integer-part>*, that shows 282 belongs to the set of sequences denoted by *<integer-part>* in the following grammar:

<real-number> ::= *<integer-part>* . *<fraction>*

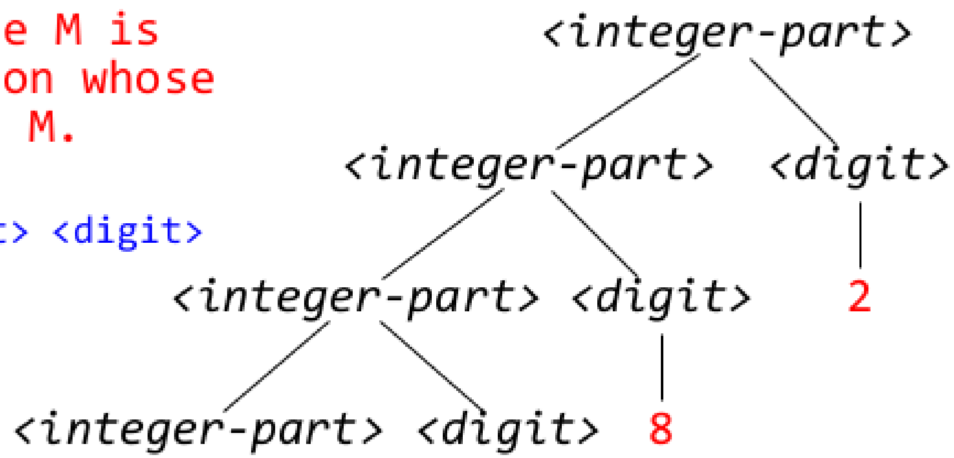
<integer-part>* ::=** *<empty>* | ***<integer-part> <digit>

<fraction> ::= *<digit>* | *<digit>* *<fraction>*

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

The left-to-right sequence of children of any internal node M is the right side of a production whose left side is the nonterminal M.

Using production:
<integer-part> ::= <integer-part> <digit>



Let's draw a parse tree, whose root is *<integer-part>*, that shows 282 belongs to the set of sequences denoted by *<integer-part>* in the following grammar:

<real-number> ::= *<integer-part>* . *<fraction>*

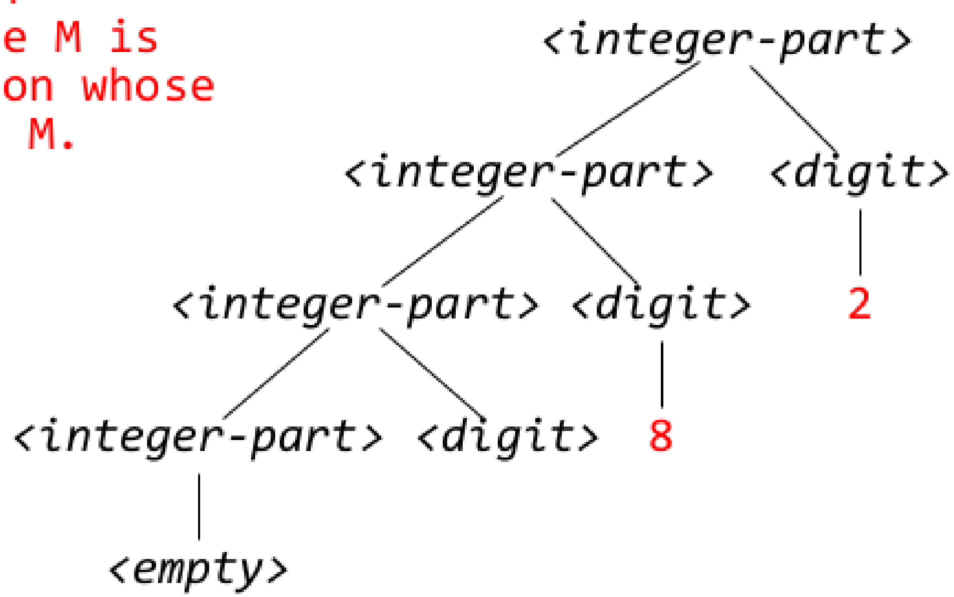
<integer-part>* ::= *<empty> | *<integer-part>* *<digit>*

<fraction> ::= *<digit>* | *<digit>* *<fraction>*

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

The left-to-right sequence of children of any internal node M is the right side of a production whose left side is the nonterminal M.

Using production:
<integer-part> ::= <empty>

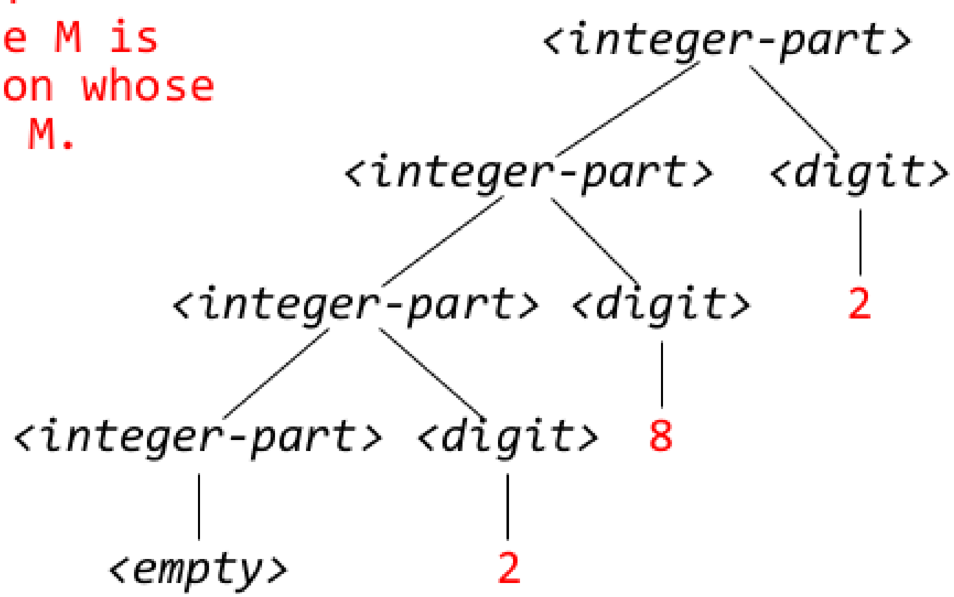


Let's draw a parse tree, whose root is *<integer-part>*, that shows 282 belongs to the set of sequences denoted by *<integer-part>* in the following grammar:

<real-number> ::= *<integer-part>* . *<fraction>*
<integer-part> ::= *<empty>* | *<integer-part>* *<digit>*
<fraction> ::= *<digit>* | *<digit>* *<fraction>*
***<digit>* ::=** 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

The left-to-right sequence of children of any internal node M is the right side of a production whose left side is the nonterminal M.

Using production:
<digit> ::= 2



Let's draw a parse tree, whose root is *<integer-part>*, that shows 282 belongs to the set of sequences denoted by *<integer-part>* in the following grammar:

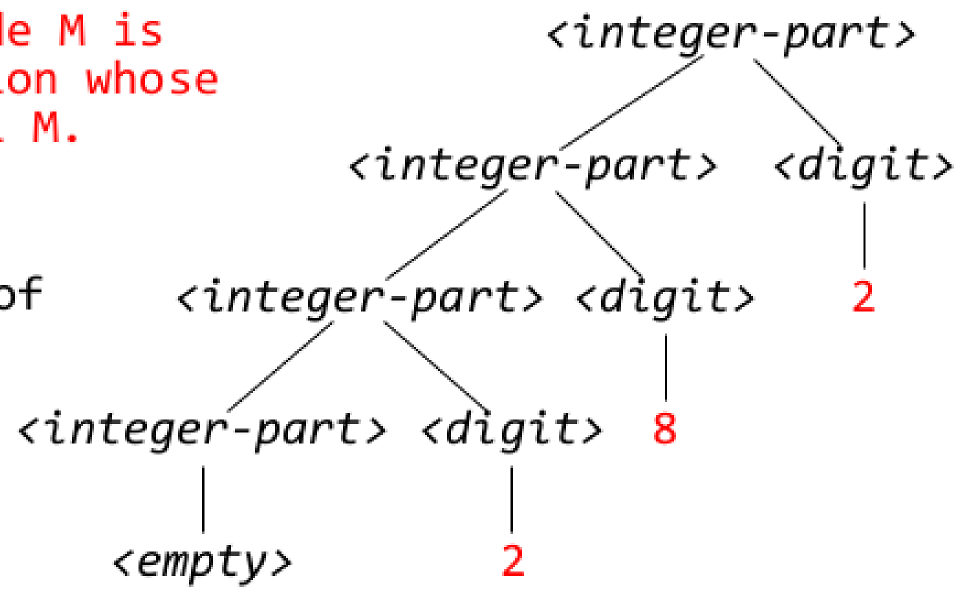
<real-number> ::= *<integer-part>* . *<fraction>*
<integer-part> ::= *<empty>* | *<integer-part>* *<digit>*
<fraction> ::= *<digit>* | *<digit>* *<fraction>*
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

The left-to-right sequence of children of any internal node M is the right side of a production whose left side is the nonterminal M.

Using production:
<digit> ::= 2

The left-to-right sequence of leaves that are not <empty> is 282, as required.

So the parse tree is complete!



RECALL:

The set of sequences of terminals denoted by the **starting nonterminal** of a grammar is called the ***language*** *generated by* (or ***language*** *of*) that grammar.

So a sequence of terminals $t_1 \ldots t_k$ belongs to the language of a grammar *if and only if* …

RECALL:

The set of sequences of terminals denoted by the **starting nonterminal** of a grammar is called the ***language*** *generated by* (or ***language*** *of*) that grammar.

So a sequence of terminals $t_1 \ldots t_k$ belongs to the language of a grammar *if and only if* there is a **parse tree**, whose root is the **starting nonterminal, that generates** $t_1 \ldots t_k$.

RECALL:

The set of sequences of terminals denoted by the **starting nonterminal** of a grammar is called the ***language*** *generated by* (or ***language*** *of*) that grammar.

So a sequence of terminals $t_1 \ldots t_k$ belongs to the language of a grammar *if and only if* there is a **parse tree**, whose root is the **starting nonterminal**, **that generates** $t_1 \ldots t_k$.

Unless otherwise indicated, the term ***parse tree*** means *parse tree <u>whose root is the **starting nonterminal**</u>*.

So we can simply say:

RECALL:

The set of sequences of terminals denoted by the **starting nonterminal** of a grammar is called the ***language*** *generated by* (or ***language*** *of*) that grammar.

So a sequence of terminals $t_1 \ldots t_k$ belongs to the language of a grammar *if and only if* there is a **parse tree**, whose root is the **starting nonterminal**, **that generates** $t_1 \ldots t_k$.

Unless otherwise indicated, the term ***parse tree*** means *parse tree* <u>*whose root is the **starting nonterminal***</u>.

So we can simply say:

- A sequence of terminals $t_1 \ldots t_k$ belongs to the language of a grammar *if and only if* there is a **parse tree that generates** $t_1 \ldots t_k$.

Below is a parse tree that shows 282.83 belongs to the language of this grammar:

```
 <real-number> ::= <integer-part> . <fraction>
<integer-part> ::= <empty> | <integer-part> <digit>
    <fraction> ::= <digit> | <digit> <fraction>
       <digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Below is a parse tree that shows 282.83 belongs to the language of this grammar:

<real-number> ::= *<integer-part>* . *<fraction>*
<integer-part> ::= *<empty>* | *<integer-part>* *<digit>*
<fraction> ::= *<digit>* | *<digit>* *<fraction>*
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9