A grammar can only have ***finitely*** many productions. However, any EBNF rule can be translated into an equivalent ***finite*** set of BNF productions as follows.

Working from the inside outwards, eliminate all occurrences of (...), [...], and { ... }:

- Replace each $(x_1 \mid \ldots \mid x_k)$ with a new nonterminal (D, say) that is defined by these k productions:

  D ::= $x_1 \mid \ldots \mid x_k$

- Replace each $[x_1 \mid \ldots \mid x_k]$ with a new nonterminal (D, say) that is defined by these k+1 productions:

  D ::= *<empty>* | $x_1 \mid \ldots \mid x_k$

- Replace each $\{x_1 \mid \ldots \mid x_k\}$ with a new nonterminal (D, say) that is defined by these k+1 productions:

  D ::= *<empty>* | $Dx_1 \mid \ldots \mid Dx_k$

Here k may be 1. Thus { *Digit* } can be replaced with a new nonterminal (*DigitSeq*, say) that is defined by:

DigitSeq ::= *<empty>* | *DigitSeq Digit*

Example: We now use the above method to translate

Expr ::= [+ | -] Term {(+ | -) Term} (*)

into a finite set of BNF productions.

1. First, replace (+ | -) with a nonterminal Op defined by: Op ::= + | -
   (*) becomes: Expr ::= [+ | -] Term {Op Term} (**)
2. Next, replace {Op Term} with a nonterminal Rest defined by: Rest ::= <*empty*> | Rest Op Term
   (**) becomes: Expr ::= [+ | -] Term Rest (***)
3. Finally, replace [+ | -] with a nonterminal OptSign defined by OptSign ::= <*empty*> | + | -
   (***) becomes: Expr ::= OptSign Term Rest

The result is the following set of 8 BNF productions:

Expr ::= OptSign Term Rest

OptSign ::= <*empty*> | + | -

Rest ::= <*empty*> | Rest Op Term

Op ::= + | -

While the above method always works, it will often ***not*** find a simplest finite set of grammar productions that is equivalent to the given EBNF rule!

For example, here is a simpler set of grammar productions that is equivalent to the EBNF rule

```
      Expr ::= [+ | -] Term {(+ | -) Term}
```

considered above:

```
  Expr ::=   Term
          |  + Term
          |  - Term
          |  Expr + Term
          |  Expr - Term
```