05 - EBNF (Extended BNF) and Syntax Diagrams

EBNF is the same as BNF, with three additional meta-symbols:

- { } which indicates *0 or more*
- [ ] which indicates *optional*
- (… | … | …) which indicates sub-alternatives

<u>EBNF has exactly the same expressive power as BNF</u>.
But it is more convenient for many applications.

Converting from BNF to EBNF must be done precisely:

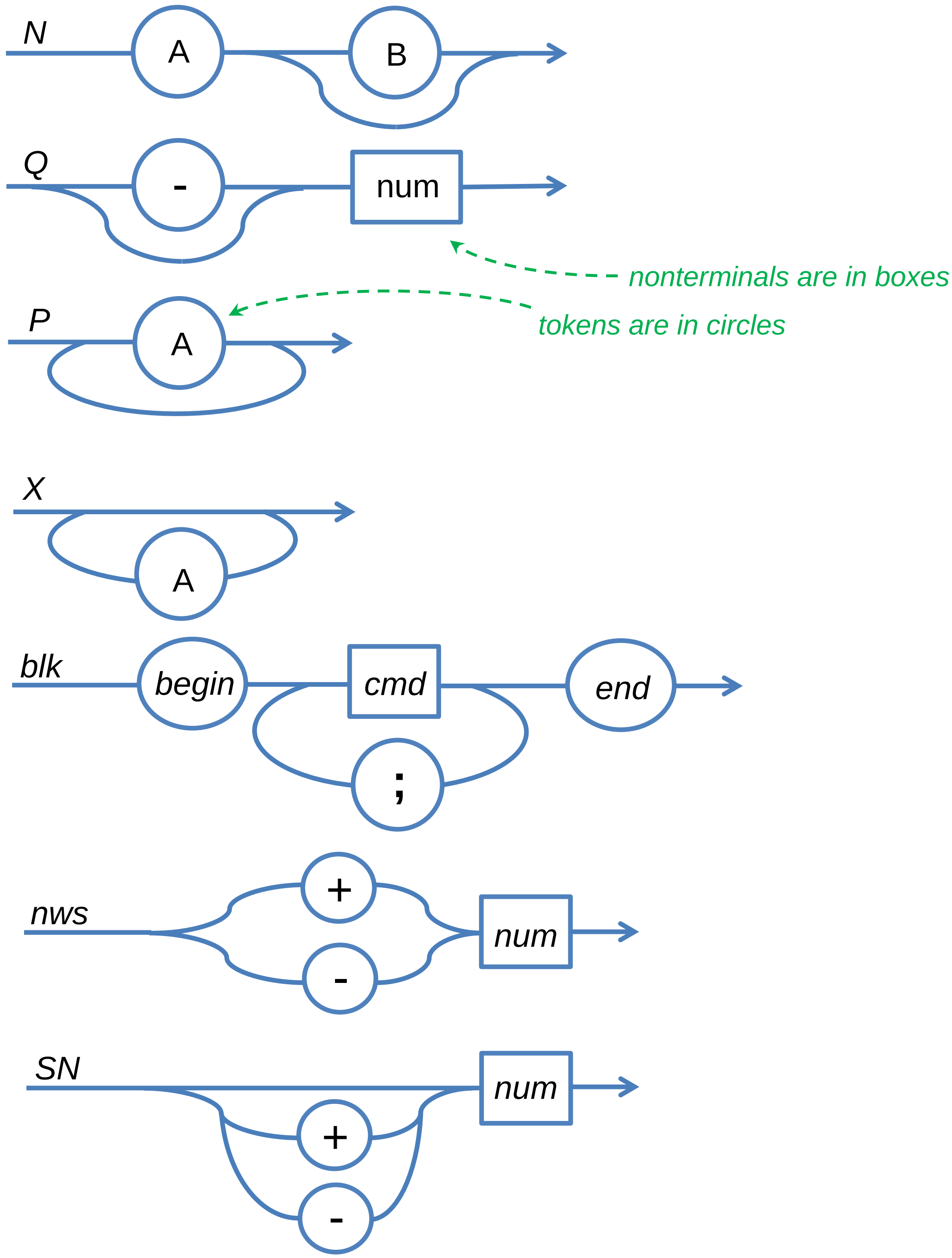
<table>
<tr><th>BNF</th><th>EBNF</th></tr>
<tr><td><N> ::= A | AB</td><td><N> ::= A [B]</td></tr>
<tr><td><Q> ::= -<num> | <num></td><td><Q> ::= [-] <num></td></tr>
<tr><td><P> ::= <P> A | A</td><td><P> ::= A { A }</td></tr>
<tr><td><X> ::= <X> A | ϵ</td><td><X> ::= { A }</td></tr>
<tr><td><blk> ::= begin <sts> end
<sts> ::= <cmd> | <cmd> ; <sts></td><td><blk> ::= begin <cmd> {; <cmd>} end</td></tr>
<tr><td><nws> ::= +<num> | -<num></td><td><nws> ::= (+|-) <num></td></tr>
<tr><td><SN> ::= +<num> | -<num> | <num></td><td><SN> ::= [(+|-)] <num></td></tr>
</table>

Some things to notice about the conversions to EBNF:

- most "or"'s (**|**) have been removed, reducing the number of rules,
- redundant items are removed when all they do is specify options,
- most recursion has been removed and replaced with { } loops, and
- occurances of the null string (**ϵ**) have been removed.

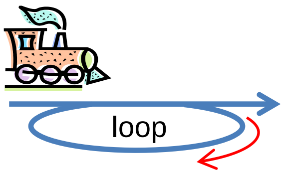
Conversion to EBNF makes it easier to draw Syntax Diagrams.
Later, we will use the Syntax Diagrams to write a recursive-descent parser.

Syntax Diagrams, sometimes called "Railroad Tracks", are graphical representations of EBNF production rules. Here are syntax diagrams for each of the examples on the previous page:

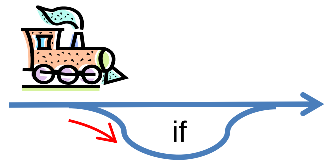


It can be helpful to imagine train tracks, to help in drawing them correctly:

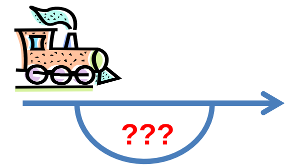
- Control structures (curves and switches) should be very clear:



OK

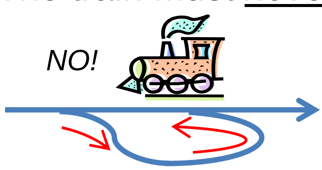


OK



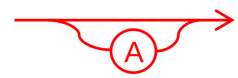
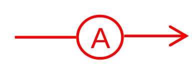
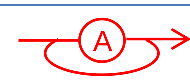
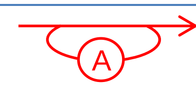
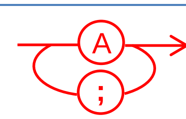
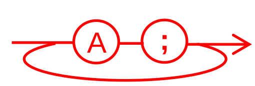
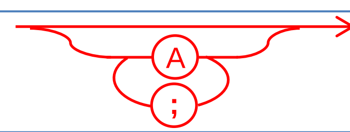
NOT LIKE THIS!
(intention must be clear)

- The train must *__never__* "reverse directions":



Although real trains can do this, computers never run the code backwards!!

There are some common structures in programming languages.
Here is the correct way to draw them in BNF, EBNF, and Syntax Diagrams:

	BNF	EBNF	Syntax Diagram
A is optional	M : : = xxAxx | xxxx	M : : = xx[A]xx	
A is required	M : : = xxAxx	M : : = xxAxx	
1 or more of A	M : : = MA | A	M : : = A { A }	
0 or more of A	M : : = MA | ∈	M : : = { A }	
1 or more of A with separators	M : : = M ; A | A	M : : = A { ; A }	
1 or more of A with terminators	M : : = MA; | A;	M : : = A ; { A ; }	
0 or more of A with separators	M : : = H | ∈ H : : = H ; A | A	M : : = [A { ; A }]	
0 or more of A with terminators	M : : = MA; | ∈	M : : = { A ; }