BNF grammars (1/3)

BNF grammars offer concise language specifications.

$$S ::= D \mid DS$$
$$D ::= 0 \mid 1$$

It consists of:

- a set of non-terminals and a set of terminals
- four productions:
  - "a S may be replaced by D"
  - "a S may be replaced by DS"
  - "a D may be replaced by 0"
  - "a D may be replaced by 1"
- A (non-terminal) start symbol S, typically written first.

BNF grammars (2/3)

BNF grammars offer concise language specifications.

$$S ::= D \mid DS$$
$$D ::= 0 \mid 1$$

They do so by expressing a set of derivable strings.

Recipe: Repeatedly replace a non-terminal by a right-hand-side until there's only terminals left:

$$S \to D \to 0$$
$$S \to DS \to 0S \to 0D \to 01$$
$$S \to DS \to 1S \to 1DS \to 10S \to 10D \to 101$$

Q: Which language does this grammar represent?

BNF grammars (2/3)

BNF grammars offer concise language specifications.

$$S ::= D \mid DS$$
$$D ::= 0 \mid 1$$

They do so by expressing a set of derivable strings.

Recipe: Repeatedly replace a non-terminal by a right-hand-side until there's only terminals left:

$$S \to D \to 0$$
$$S \to DS \to 0S \to 0D \to 01$$
$$S \to DS \to 1S \to 1DS \to 10S \to 10D \to 101$$

Q: Which language does this grammar represent?

Exercise: alter the grammar to yield base-10 numbers

BNF grammars (3/3)

BNF is short for 'Backus-Naur Form' named after John Backus and Peter Naur.

A BNF can express a 'Context-Free Language'. These are strictly more powerful than, e.g., regular expressions.

Most language specifications today come with a BNF.

Q: Which languages do the following grammars define?

$$S ::= 0 \mid 01 \mid 01S$$

$$S ::= ab \mid aSb$$

$$S ::= e \mid S + S \mid S - S$$

$$S ::= s \mid s\texttt{;}\, S \mid \{S\}$$