### Context-Free Grammars (CFGs)

A CFG is specified by:

* A finite set of **terminal** symbols.
* A finite set of **non-terminal** symbols (disjoint from the terminals).
* A finite set of **productions** of the form:  
   where:
  + is the non-terminal symbol.
  + is the sequence (possibly the null sequence) of terminals and non-terminals.
* A **starting** non-terminal symbol.

### Translation Grammars (TGs)

A TG is a CFG where the set of terminals is partitioned into:

1. A set of **input symbols**.
2. A set of **action symbols**.

The strings in the language specified by the TG are **activity sequences**.

A CFG can be converted into a TG by inserting **action symbols** at appropriate locations in the productions.

A TG in which all action symbols specify **output routines** is called a **string translation grammar**.

### Attributed Table Grammars (ATGs)

An ATG is a TG where:

* Each input, non-terminal and action symbol has an associated **finite set of inputs**.
* Each non-terminal and action symbol attribute is classified as either:
  1. **Inherited** - Evaluated from the **top down** or **across** the tree.
  2. **Synthesized** - Evaluated from the **bottom up**.
* For each occurance of an **inherited attribute** on the **RHS** of a given production, there is an associated rule which describes how to **compute a value** for that attribute, as a function of certain other attributes of symbols occurring in the LHS or RHS of the production.
* An **initial value** is specified for each inherited attribute of the **starting** non-terminal.
* For each occurance of a **synthesized attribute** on the **LHS** of a given production, there is an associated rule which describes how to **compute a value** for that attribute, as a function of certain other attributes of symbols occurring in the LHS or RHS of the production.
* For each **synthesized action-symbol attribute**, there is an associated rule which describes how to **compute a value** for that attribute as a function of certain other attributes of the action symbol.

### S-Grammars

S-Grammars have the following properties:

* Every production **begins** with a **terminal** symbol.
* All productions for the **same** non-terminal **start with a different terminal** symbol.

### Q-Grammars

Q-Grammars have the following properties:

* The RHS of each production is **either empty** () or **begins with a terminal** symbol.
* Productions for the **same** LHS non-terminal symbol have **disjoint selection sets**.

### LL(1) Grammars

A grammar is LL(1) if and only if all productions for the same non-terminal symbol have **disjoint selection sets**.

Given a production , where is a string of terminal and non-terminal symbols:

* If is **non-nullable**:
* Else if is **nullable**:

### L-Attributed Grammars

A grammar is L-Attributed if and only if:

* For each attribute-evaluation rule associated with the **inherited** attribute of a symbol in the **RHS** of a production, each argument of that rule is either:
  1. An inherited attribute of the LHS.
  2. An arbitrary attribute of some RHS symbol appearing to the left of a given symbol.

(i.e. You can only inherit from the left, since we evaluate from left to right)

* For each attribute-evaluation rule associated with the **synthesized** attribute of a symbol in the **LHS** of a production, each argument of that rule is either:
  1. An inherited attribute of the LHS.
  2. An arbitrary attribute of some RHS symbol.
* For each attribute-evaluation rule associated with the **synthesized** attribute of an action symbol, each argument of the rule is:
  1. An inherited attribute of the given action symbol.

These three conditions ensure that the attribute-evaluation rules can be **evaluated from left to right**.