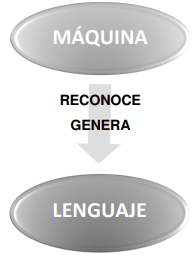
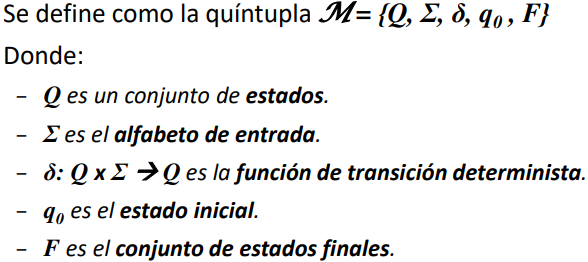
Relacion Lenguaje-Automata



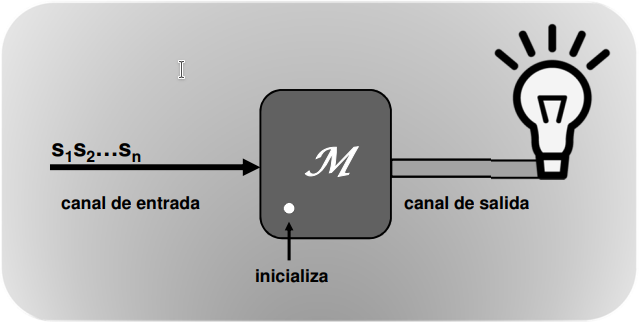
* Están dadas en el estudio de autómatas aceptadores, generadores y traductores. Es relevante:
  + El estudio matemático de estas maquinas (autómatas):
    - Capacidades:
      * Dominio de elementos que maneja, con que elementos va a trabajar cada modelo.
      * Me va a interesar mas en la parte de aplicación.
      * Ej cada maquina que es lo que puede hacer.
    - Síntesis:
      * Distintos tipos de maquinas se van a poder interconectar para poder realizar una determinada tarea.
      * Me va a interesar mas en la parte del estudio.
      * Ej para un traductor se vinculan distintos autómatas de diferentes maquinas que hacen diferentes tareas, esto seria el estudio de la síntesis.
  + La aplicación de esta teoría en el diseño de lenguajes, editores, compiladores, interpretes:
    - Expresar el diseño de un compilador como la interconexión de varios de estos modelos.
    - Implementar o simular estos modelos con programas

Aceptadores

* Autómata diseñado fundamentalmente para aceptar o rechazar cadenas.
* Va a dar una respuesta, si pertenece o no al lenguaje.

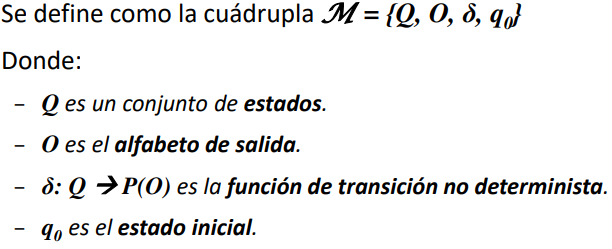


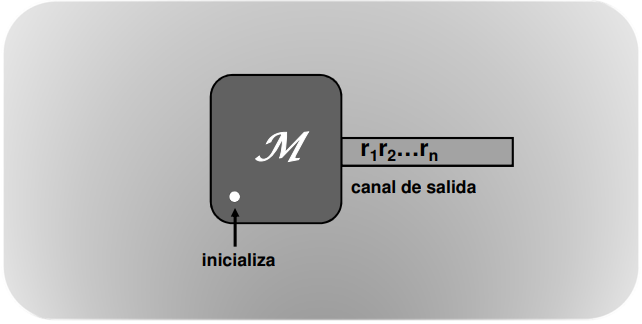
Funcion de transición deterministica = misma entrada, mismo retorno



Generadores

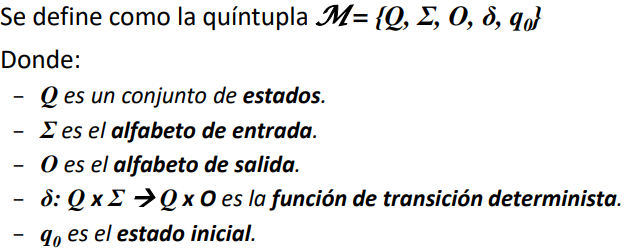
* Autómata que genera símbolos de un alfabeto finito, cuando es encendido.

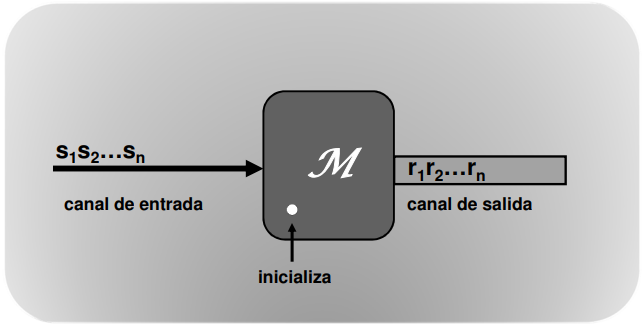




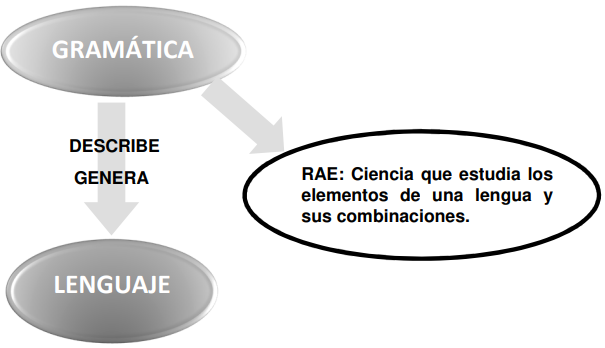
Traductores

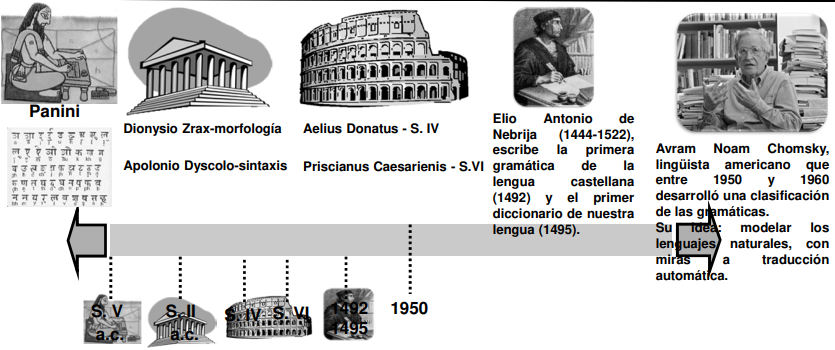
* Deterministico.
* Muy útil en el diseño de compiladores





Relación Lenguaje-Gramática





Panini: Traducir a través de unas 4000 reglas el Sánscrito. Desarrollos gramaticales de este lingüista se siguen usando hasta hoy en día.

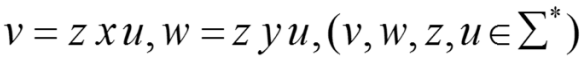
Dionysio Zrax y Apolonio Dyscolo: ver como se combinaban los símbolos del alfabeto y ver como se como esas palabras se podían combinar en una oración.

Aelius Donatus y Priscianus Caesarienis: trabajaron con cuestiones de gramática y demás.

En el periodo del renacimiento era muy importante todo lo que era leer y escribir. Se desarrollaba mucho la dialéctica.

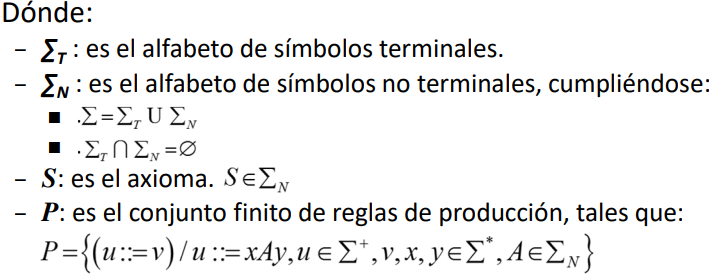
Gramáticas Formales

* ¿Cómo definir las palabras que pertenecen a un lenguaje?
  + Enumerando el conjunto de palabras.
  + Descripción informal de palabras.
  + Descripción basada en operadores de palabra
* Es necesario un formalismo para definir lenguajes
  + **Gramáticas Formales:**
    - Descripciones estructurales de las sentencias de los lenguajes.
    - Definen un lenguaje mediante reglas de producción.
* La **sintaxis** de un lenguaje de programación es una definición de lo que constituye un programa gramaticalmente válido en dicho lenguaje. Verifica que este bien escrito
* Queda especificada por un conjunto de reglas:
  + Reglas léxicas: Especifican el conjunto de caracteres que constituyen el alfabeto del lenguaje y la forma en que estos se combinan para formar palabras validas. Verifica también si el carácter pertenece al alfabeto.
  + Reglas sintácticas: especifican la forma en que se construyen las estructuras validas de un programa. Ej. Palabras bien formadas conformando una oracion de manera correcta (sujeto, predicado, etc.).
* Producción o regla (x::=y):
  + Es un par ordenado de palabras.
  + Si se encuentra x como parte de cualquier palabra v, se puede sustituir x por y en v.
    - **x** es la cabeza o parte izquierda de la producción.
    - **y** es el cuerpo o parte derecha de la producción.
* Derivación directa
  + Aplicación de una producción (x::=y) a una palabra v para convertirla en otra palabra w, donde:



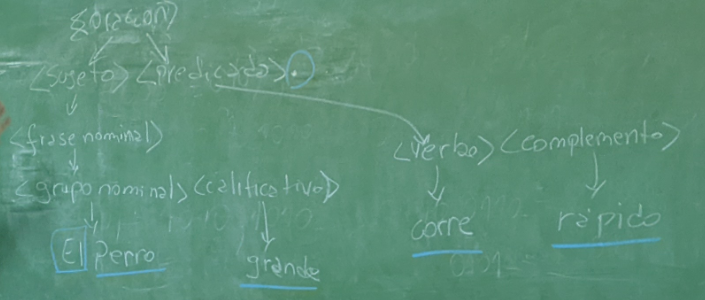
* Derivación
  + Aplicación de una secuencia de producciones a una palabra
* Derivación más a la izquierda:
  + Se utiliza en cada derivación directa la producción aplicada a los símbolos más a la izquierda de la palabra.
* Derivación más a la derecha:
  + Se utiliza en cada derivación directa la producción aplicada a los símbolos más a la derecha de la palabra.
* Ejemplo en castellano:
  + La sintaxis del castellano se define mediante reglas:
    1. Una oración consta de sujeto y predicado y termina con un punto.
    2. Un sujeto es una frase nominal.
    3. Una frase nominal es un grupo nominal seguido de un calificativo (no siempre está presente).
    4. El predicado puede ser un verbo o un verbo y su complemento.
* Una gramática **G** se define como una cuádrupla:





* En la parte izquierda de mis reglas de producción tiene que haber algo para poder aplicar reglas de derivación, para obtener palabras validas.
* Forma sentencial: x es forma sentencia si existe una derivación desde el axioma hasta esa palabra, .
* Sentencia: x es sentencia si es forma sentencial y todos sus símbolos pertenecen al alfabeto de símbolos terminales





* Lenguaje generado por una gramática L(G): es el conjunto de todas las sentencias de la gramática



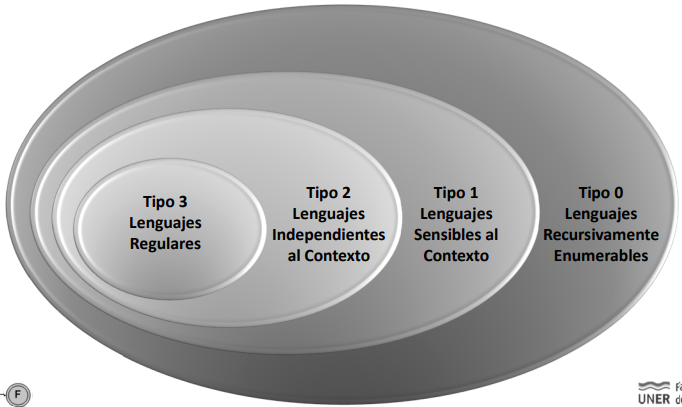
* Equivalencia de gramáticas:
  + Dos gramáticas G1 y G2 son equivalentes si se verifica que:



* Distinta estructura pero definen el mismo lenguaje.
* Por obvias razones deben compartir el alfabeto.
* Regla compresora:
  + Es aquella cuya parte derecha está formada por menos símbolos que la parte izquierda.



Jerarquía de Chomsky



* Las gramáticas se diferencian entre si por sus reglas de producción.
* Tipos:
  + Gramáticas Tipo 0 (Sin restricciones):
    - En la parte izquierda debe existir aunque sea un símbolo no terminal.
    - Respecto de las partes derechas de las producciones no hay ningún tipo de restricción.



* Gramáticas Tipo 1 (Dependientes o Sensibles al Contexto):
  + La parte izquierda y derecha de las producciones tienen que tener una parte común.
  + Solo se admite como regla compresora S::=



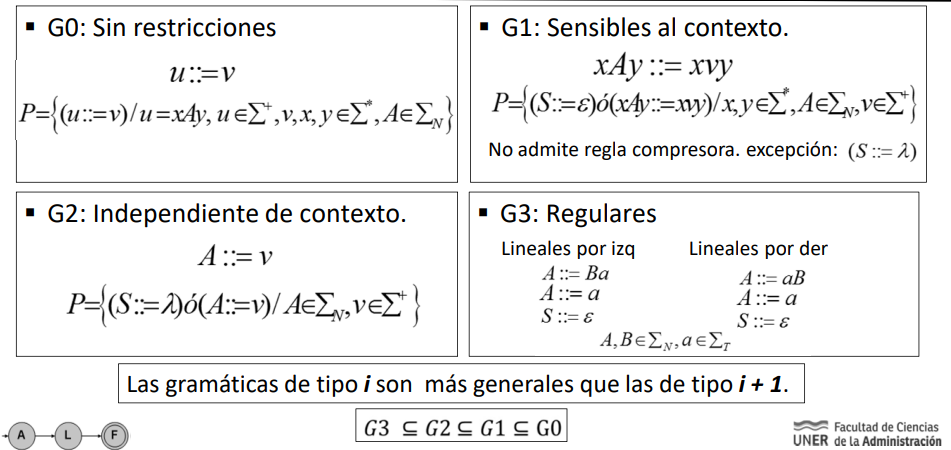
* + Particularidad



* Gramáticas Tipo 2 (Independientes o Libres de Contexto):
  + Permiten la descripción de la mayoría de los lenguajes de programación.
  + La parte izquierda de las producciones sólo puede tener un símbolo no terminal.
  + Las producciones son de la forma:

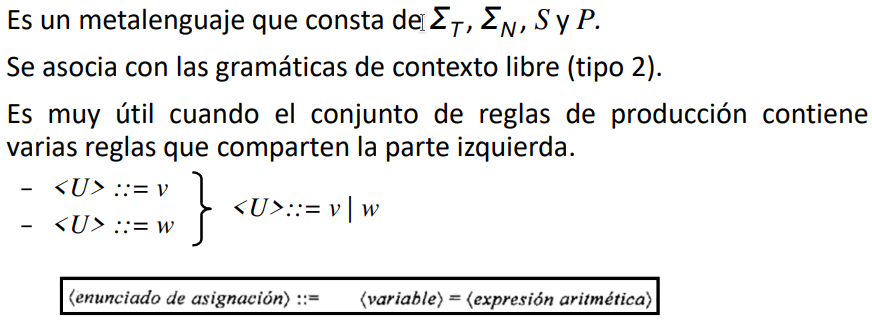


* Gramáticas Tipo 3 (Regulares o Lineales)
  + - * Son las gramáticas mas restrictivas.
      * Las producciones pueden ser:
        + Lineales por izquierda:
* Lineales por derecha:



Notaciones Formales

* BNF (Backus-Naur Form)



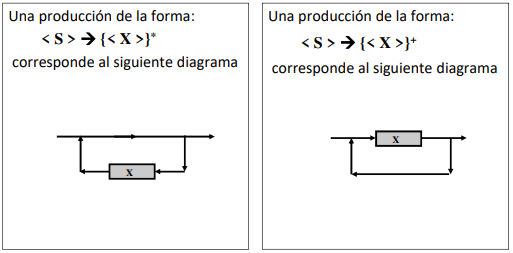
* EBNF (Backus-Naur Form Extended):
  + Mejorar legibilidad del BNF.
  + Puede reducir el tamaño de la especificación.
  + Se pueden confundir los símbolos del metalenguaje con los del lenguaje sujeto.
  + Extensiones comunes:
    - [ ] (corchetes): para indicar elementos opcionales.
    - { } (llaves): para indicar repeticiones en la parte derecha de la regla.
      * \* (asterisco): cero o mas iteraciones.
      * + (más): una o mas iteraciones.
    - ( ) (paréntesis): para indicar alternativa en la parte derecha de la regla.
      * | (pipe): para separar las alternativas.
* Grafos o Diagramas Sintácticos:
  + Son gráficos que se utilizan para representar las reglas de una gramática BNF o EBNF.
  + Cada regla esta determinada por un camino:
    - Los símbolos **no terminales** se representan con rectángulos.



* + - Los símbolos **terminales** se representan con círculos.







* Arboles de Derivación:
  + - Representación gráfica de las derivaciones de una gramática.
    - Se utilizan en la construcción de compiladores.
    - Construcción:
      * La raíz del árbol se etiqueta con el axioma de la gramática.
      * Los nodos hojas son símbolos terminales de la gramática.
      * Los nodos intermedios son símbolos no terminales de la gramática.

Ambigüedad

* Existe más de una forma de generar una palabra a partir del axioma de una gramática.
* Puede surgir a varios niveles:
  + - Sentencia: si tiene más de una derivación o árbol de derivación.
    - Gramática: si tiene al menos una sentencia ambigua.
    - Lenguaje: si existe una gramática ambigua que lo genera. Si todas las gramáticas que lo generan son ambiguas, el lenguaje es inherentemente ambiguo.

Recursividad

* Es la definición de un concepto que utiliza a ese mismo concepto en la definición.
* Niveles de recursividad:
  + **Producción recursiva:** si el símbolo no terminal aparece en los dos lados de la producción.





* + **Gramática recursiva:** si existe al menos una producción recursiva en su conjunto de producciones.

Resumen

