Slide 1

TIPOS DE REGLAS GRAMATICALES (CHOMSKY)

0 - no restringidas	$\alpha \to \beta \quad \alpha \neq \varepsilon$	$abc \to bA$
0'	$\alpha A \alpha' \to \beta$	$aABc \to bbC$
- con estructura de frase	$\alpha A \alpha' \to \alpha \beta \alpha'$	$aAAC \to aAC$ $A \to \varepsilon$
1 - dependientes de contexto	$\alpha A \alpha' \to \alpha \beta \alpha'$ $\beta \neq \varepsilon$	$AAC \to baAC$ $aAAC \to aAaAC$
2 -	$A \to \beta \quad \beta \neq \varepsilon$	$A \to bABa\,;\, A \to B$
3 i -	$A \to \beta$ $\beta \in \Sigma_A \Sigma_T \cup \Sigma_T$	$A \to Aa$ $A \to a$
3 d -	$A \to \beta$ $\beta \in \Sigma_T \Sigma_A \cup \Sigma_T$	$A \to aB$ $A \to a$

Tipo 3 $\Rightarrow$ Tipo 2 $\Rightarrow$ Tipo 1 $\Rightarrow$ Tipo e.f. $\Rightarrow$ Tipo 0' $\Rightarrow$ Tipo 0

Slide 2

OTROS TIPOS DE REGLAS GRAMATICALES

no contractivas	$\alpha \to \beta$ $\lvert\alpha\rvert \leq \lvert\beta\rvert$	$aBC \to ACb$
independientes de contexto	$A \to \beta$	$A \to aAB$ $A \to \varepsilon$
regulares por la izquierda	$A \to \beta$ $\beta \in \Sigma_A \Sigma_T \cup \Sigma_T \cup \{\varepsilon\}$	$A \to Ba$ $A \to \varepsilon$
regulares por la derecha	$A \to \beta$ $\beta \in \Sigma_T \Sigma_A \cup \Sigma_T \cup \{\varepsilon\}$	$A \to aA$ $A \to a$

Tipo 1 $\Rightarrow$ no contractiva

Tipo 2 $\Rightarrow$ independiente de contexto

Tipo 3 $\Rightarrow$ regular

Gramática de tipo **t**: todas sus reglas son de tipo **t**

Slide 3

TIPOS DE LENGUAJES

L es de tipo **t** si existe una gramática de tipo **t** que genera $L-\{\varepsilon\}$

Si L es de tipo **t**, entonces $L-\{\varepsilon\}$ y $L\cup\{\varepsilon\}$ son de tipo **t**

L de tipo 3 $\Leftrightarrow$ L regular

L de tipo 2 $\Leftrightarrow$ L independiente de contexto

Si G es no contractiva, existe una gramática de tipo 1 equivalente.

L de tipo 1 $\Leftrightarrow$ $L-\{\varepsilon\}$ generable por una gramática no contractiva

Si G es de tipo 0, existe una gramática equivalente con e. de frase.

L de tipo 0 $\Leftrightarrow$ L generable por una gramática con e. de frase

Slide 4

Si G es no contractiva, existe una gramática de tipo 1 equivalente:

$r:$	$X_1X_2X_3\ldots X_{p-1}X_p$	$\rightarrow$	$Y_1Y_2Y_3\ldots\ldots Y_q$	$p\leq q$
$r_1:$	$X_1X_2X_3\ldots X_{p-1}X_p$	$\rightarrow$	$Z_rX_2X_3\ldots X_{p-1}X_p$	
$r_{2_2}:$	$Z_rX_2X_3\ldots X_{p-1}X_p$	$\rightarrow$	$Z_rY_2X_3\ldots X_{p-1}X_p$	
$r_{2_3}:$	$Z_rY_2X_3\ldots X_{p-1}X_p$	$\rightarrow$	$Z_rY_2Y_3\ldots X_{p-1}X_p$	
$\ldots$	$\ldots\ldots\ldots$	$\ldots$	$\ldots\ldots\ldots$	
$r_{2_{p-1}}:$	$Z_rY_2Y_3\ldots X_{p-1}X_p$	$\rightarrow$	$Z_rY_2Y_3\ldots Y_{p-1}X_p$	
$r_{2_p}:$	$Z_rY_2Y_3\ldots Y_{p-1}X_p$	$\rightarrow$	$Z_rY_2Y_3\ldots Y_{p-1}Y_p\ldots Y_q$	
$r_3:$	$Z_rY_2Y_3\ldots\ldots\ldots\ldots Y_q$	$\rightarrow$	$Y_1Y_2Y_3\ldots\ldots\ldots\ldots Y_q$	

Si X_i es terminal, en todas las reglas se cambia X_i por X_i' y se añade

$r_{4_i}:\quad X_i'\quad\rightarrow\quad X_i$

Slide 5

S	$\rightarrow$	$abMS$	abM	$\rightarrow$	$baab$	$S \rightarrow a$	$M \rightarrow b$
			$\underline{a}bM$	$\rightarrow$	$\underline{Z}bM$		
			$Z\underline{b}M$	$\rightarrow$	$Z\underline{a}M$		
			$Za\underline{M}$	$\rightarrow$	$Za\underline{ab}$		
			$\underline{Z}aab$	$\rightarrow$	$\underline{b}aab$		
S	$\rightarrow$	$ABMS$	ABM	$\rightarrow$	ZBM	$S \rightarrow A$	$M \rightarrow B$
A	$\rightarrow$	a	ZBM	$\rightarrow$	ZAM		
B	$\rightarrow$	b	ZAM	$\rightarrow$	$ZAAB$		
			$ZAAB$	$\rightarrow$	$BAAB$		

Slide 6

MÁQUINAS DE TURING

$M = (\Sigma_E, Q, \Gamma, q_1, \hbar, f, F)$ donde

Σ_E	:	alfabeto *de entrada*
Γ	:	alfabeto *de la cinta*
Q	:	conjunto de *estados*, $finito$
$q_1 \in Q$	:	estado *inicial*
$\hbar \in \Gamma - \Sigma_E$	:	símbolo *blanco*
f	:	$Q \times \Gamma \rightarrow Q \times \Gamma \times \{\leftarrow, \rightarrow\}$
		función *parcial* de transición
$F \subseteq Q$	:	estados *finales* o *de aceptación*

No determinista: $f : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{\leftarrow, \rightarrow\})$

Slide 7

configuración: $(q, \alpha_1 \underline{A} \alpha_2) \in (Q, \Gamma^* \Gamma \Gamma^*)$, que especifica el estado, el contenido significativo[a] de la cinta, y el símbolo sobre el que se encuentra la cabeza

configuración *compacta*: cadena de $\Gamma^* Q \Gamma^*$: $\alpha_1 q A \alpha_2$

configuración inicial: $(q_1, \underline{A}x)$ compacta: $q_1 A x$

movimiento:

Si $f(q, A) = (q', B, \rightarrow)$:

$(q, \alpha_1 I \underline{A} D \alpha_2) \vdash (q', \alpha_1 I B \underline{D} \alpha_2)$ $\quad$ $\alpha_1 I q A D \alpha_2 \vdash \alpha_1 I B q' D \alpha_2$

Si $f(q, A) = (q', B, \leftarrow)$:

$(q, \alpha_1 I \underline{A} D \alpha_2) \vdash (q', \alpha_1 \underline{I} B D \alpha_2)$ $\quad$ $\alpha_1 I q A D \alpha_2 \vdash \alpha_1 q' I B D \alpha_2$

Si $f(q, A)$ no está definida, la M.T. no se mueve (se para)

[a] no blancos, más el blanco sobre el que se encuentre la cabeza si es el caso

Slide 8

Posibilidades de computación para una M.T., M:

- Partiendo de (q_1, x) $\quad x \in \Sigma_E^*$
  - M no se para $((q_1, x) \vdash^* \infty)$
  - M se para en la configuración (q, β):
    Ha transformado x en β : $f_M : \Sigma_E^* \rightarrow \Gamma^*$, (función parcial)
    - $q \in F$: acepta la cadena x
    - $q \notin F$: no acepta la cadena x
- El conjunto de cadenas de Γ^* que deja en la cinta como consecuencia de todas las computaciones en las que se para constituye un lenguaje sobre Γ : genera un lenguaje.

Slide 9

$\Gamma = \{|, \bullet, \hbar\} \quad \Sigma = \{\,|\,\}$

$q_1 = p \qquad F = \{s\}$

f :

$MT1$	$\mid$	$\bullet$	$\hbar$
$\rightarrow p$	$q\ \bullet\ \rightarrow$	$p\ \bullet\ \leftarrow$	$r\ \hbar\ \rightarrow$
q	$q\ \mid\ \rightarrow$	$q\ \bullet\ \rightarrow$	$p\ \bullet\ \leftarrow$
r		$r\ \mid\ \rightarrow$	$s\ \hbar\ \leftarrow$
(s)			

$(p, \underline{|}\,|\,) \vdash (q, \bullet \underline{|}\,) \vdash (q, \bullet\,|\,\underline{\hbar}\,)$

$\vdash (p, \bullet \underline{|}\, \bullet\,)$

$\vdash (q, \bullet \bullet \underline{\bullet}\,) \vdash (q, \bullet \bullet \bullet \underline{\hbar}\,)$

$\vdash (p, \bullet \bullet \bullet \underline{\bullet}\,) \vdash (p, \bullet \bullet \underline{\bullet}\, \bullet\,) \vdash (p, \bullet \underline{\bullet}\, \bullet \bullet\,) \vdash (p, \underline{\bullet}\, \bullet \bullet \bullet\,)$

$\vdash (p, \underline{\hbar}\, \bullet \bullet \bullet \bullet\,)$

$\vdash (r, \underline{\bullet}\, \bullet \bullet \bullet\,) \vdash (r, |\, \underline{\bullet}\, \bullet \bullet\,) \vdash (r, |\,|\, \underline{\bullet}\, \bullet\,) \vdash (r, |\,|\,|\, \underline{\bullet}\,) \vdash (r, |\,|\,|\,|\, \underline{\hbar}\,)$

$\vdash (s, |\,|\,|\,|\, \underline{|}\,) \perp$

Slide 10

- Partiendo de $(q_1, x) \quad x \in \{\,|\,\}*$

  M se para (siempre) en la configuración $(s, |^{2|x|})$:

  Ha transformado $|^n$ en $|^{2n}$: $f_M : \mathbb{N} \rightarrow \mathbb{N}$, función total que es la multiplicación por 2 (en unario)

  Como $s \in F$: acepta la cadena x (acepta todo $|^*$)

- El conjunto de cadenas de que deja en la cinta como consecuencia de todas las computaciones en las que se para constituye el lenguaje $(||)^*$: genera $(||)^*$.

Slide 11

$\Gamma = \{|, \bullet, \hbar\}$ $\quad \Sigma = \{\,|\}$

$q_1 = p$ $\quad F = \{s\}$

f :

$MT11$	$\mid$	$\bullet$	$\hbar$
$\rightarrow p_0$			$q_0 \mid \leftarrow$
q_0			$p\ \hbar\ \rightarrow$
p	$q \bullet \rightarrow$	$p \bullet \leftarrow$	$r\ \hbar\ \rightarrow$
q	$q \mid \rightarrow$	$q \bullet \rightarrow$	$p \bullet \leftarrow$
r		$r \mid \rightarrow$	$s\ \hbar\ \leftarrow$
$[s\,]$	$s \mid \leftarrow$		$p\ \hbar\ \rightarrow$

$$
\begin{aligned}
(p_0, \underline{\hbar}) &\vdash (q_0, \underline{\hbar}\,|\,) \\
&\vdash (p, \underline{|}\,) \vdash^* (s, |\underline{|}) \vdash^* (s, \underline{\hbar}||) \\
&\vdash (p, \underline{||}\,) \vdash^* (s, |||\underline{|}) \vdash^* (s, \underline{\hbar}||||) \\
&\vdash (p, \underline{||||}\,) \vdash^* (s, |||||||\underline{|}) \vdash^* (s, \underline{\hbar}|||||||) \\
&\vdash (p, \underline{|||||||}\,) \vdash \ldots
\end{aligned}
$$

Slide 12

- Partiendo de $(p_0, x) \quad x \in |^*$
  - si $x = \varepsilon$, M no se para
    si $x \neq \varepsilon$, M se para en p_0
  - si M se para (con $x \neq \varepsilon$), lo hace en la configuración (p_0, x):
    Ha transformado x en $x : f_M : \Sigma_E^* \rightarrow \Gamma^*$, (función identidad, parcial, porque sólo se evalúa para cadenas no vacías)
    - $p_0 \notin F$: no acepta ninguna cadena
- Partiendo de $(p_0, \hbar)$ la máquina no se para, pero cada vez que pasa por el estado s, en la cinta hay una cadena de $\{|^{2^n} \ / \ n \geq 0\}$

Slide 13

	0	1	$\hbar$
$\rightarrow q_1$	$q_2\ 1\ \rightarrow$	$q_3\ 0\ \leftarrow$	
q_2	$q_2\ 0\ \rightarrow$	$q_2\ 1\ \rightarrow$	$q_1\ \hbar\ \leftarrow$
q_3	$q_2\ 1\ \rightarrow$	$q_3\ 0\ \leftarrow$	$q_2\ 1\ \rightarrow$

$(q_1, \underline{0}) \quad \vdash \quad (q_2, 1\underline{\hbar}) \vdash$

$(q_1, \underline{1}) \quad \vdash \quad (q_3, \underline{\hbar}0) \vdash (q_2, 1\underline{0}) \vdash (q_2, 10\underline{\hbar})$

$(q_1, 1\underline{0}) \quad \vdash \quad (q_2, 11\underline{\hbar})$

$(q_1, 1\underline{1}) \quad \vdash \quad (q_3, \underline{1}0) \vdash (q_3, \underline{\hbar}00) \vdash (q_2, 1\underline{0}0) \vdash (q_2, 10\underline{0}) \vdash (q_2, 100\underline{\hbar})$

$(q_1, 10\underline{0}) \quad \ldots$

Slide 14

$$LR(M) := \{x \in \Sigma_E^* \ / \ (q_1, x) \vdash^* (q, \alpha) \perp \quad q \in F\}$$

L es **recursivo** si es $LR(M)$ para alguna M.T. que se pare ante cualquier entrada.

$$LG(M) := \{\alpha \in \Gamma^* \ / \ (q_1, \hbar) \vdash^* (q, \alpha) \quad q \in F\}$$

L es **recursivamente numerable** si es $LG(M)$ para alguna M.T.

$$f_M : \Sigma_E^* \rightarrow \Gamma^* : f_M(x) := \alpha \ / \ (q_1, x) \vdash^* (q, \alpha) \perp$$

f es **computable** si es f_M para alguna M.T.

Slide 15

CONSTRUCCIÓN DE M.T.

- Composición
- Subrutinas
- Almacenamiento de información en los estados
- Cinta con varias pistas

EXTENSIONES DE M.T.

- M.T. multicinta
- Cinta multidimensional
- M.T. no determinista
- Movimiento nulo ($\leftrightarrow$)

RESTRICCIONES DE M.T.

- M.T. con cinta semiinfinita
- Alfabeto binario. Alfabeto unario

Slide 16

L es **recursivo** si existe un programa Pascal que siempre termina (y caracteriza L) de la forma

```
program ReconoceL (input, output);
var x : string:
Begin
    readln (x);
    sus cálculos, para determinar si x está o no en L
    if (ha obtenido x está en L) then
        writeln ('¡Si!')
    else
        writeln ('No')
End.
```

o lo que es equivalente: existe una función Pascal de la forma

```
function enL (x:string):boolean
(*  enL(x) = TRUE ⇔ x ∈ L  *)
```

Slide 17

L es **recursivamente numerable** si existe un programa en Pascal

```
program generaL (output);
var x : string ;
begin
     algunas operaciones (conseguir la primera x)
     while (1=1) do
       begin
         writeln (x)
         algunas operaciones (conseguir la siguiente x)
       end
end.
```

o lo que es equivalente: existen $x_1 \in L$ y

```
procedure siguienteEnL (var x : string)
(*  Pre : x ∈ L                Post: x ∈ L  *)
```

tales que $\{\texttt{siguienteEnL}^n(x_1) \ / \ n \geq 0\} = L$

Slide 18

f es **computable total** si existe una función en Pascal:

```
function f (x : string): string;
(*  Pre: x ∈ Σ*_E
Post: f = f(x)  *)
```

f es **parcialmente computable** si existe una función en Pascal:

```
function f (x : string): string;
(*  Pre: x ∈ U ⊆ Σ*_E  Post: f = f(x)  *)
```