Ambigüedad y forma normal de Chomsky

UNIVERSIDAD DE CHILE

Arturo Merino

Recuerdo / Plan

Recuerdo/Plan

- **Recuerdo:**

Recuerdo/Plan

- **Recuerdo:**
  - Equivalencia AP-GLC

Recuerdo/Plan

- **Recuerdo:**
  - Equivalencia AP-GLC
  - GLC $\rightarrow$ AP: simular las producciones por un lado.

Recuerdo/Plan

- **Recuerdo:**
  - Equivalencia AP-GLC
  - GLC $\rightarrow$ AP: simular las producciones por un lado.
  - AP $\rightarrow$ GLC: programación dinámica.

Recuerdo/Plan

- **Recuerdo:**
  - Equivalencia AP-GLC
  - GLC $\rightarrow$ AP: simular las producciones por un lado.
  - AP $\rightarrow$ GLC: programación dinámica.

- **Plan:**

Recuerdo/Plan

- **Recuerdo:**
  - Equivalencia AP-GLC
  - GLC $\rightarrow$ AP: simular las producciones por un lado.
  - AP $\rightarrow$ GLC: programación dinámica.

- **Plan:**
  - Ambigüedad

Recuerdo/Plan

- **Recuerdo:**
  - Equivalencia AP-GLC
  - GLC $\rightarrow$ AP: simular las producciones por un lado.
  - AP $\rightarrow$ GLC: programación dinámica.

- **Plan:**
  - Ambigüedad
  - Forma normal de Chomsky

Recuerdo/Plan

- **Recuerdo:**
  - Equivalencia AP-GLC
  - GLC $\rightarrow$ AP: simular las producciones por un lado.
  - AP $\rightarrow$ GLC: programación dinámica.

- **Plan:**
  - Ambigüedad
  - Forma normal de Chomsky
  - Wrap-up lenguajes libres de contexto

Ambigüedad

Ambigüedad

- Strings con derivación “única”

Ambigüedad

- Strings con derivación “única”
  - Podemos definir funciones/propiedades sobre la derivación

Ambigüedad

- Strings con derivación “única”
  - Podemos definir funciones/propiedades sobre la derivación
  - Interpretabilidad

Ambigüedad

- Strings con derivación "única"
  - Podemos definir funciones/propiedades sobre la derivación
  - Interpretabilidad

Def.

Un string w será *derivado ambiguamente* por G si tiene dos o más derivaciones por la izquierda (o dos o más árboles de derivación).

Ambigüedad

- Strings con derivación "única"
  - Podemos definir funciones/propiedades sobre la derivación
  - Interpretabilidad

Def.

Un string w será *derivado ambiguamente* por G si tiene dos o más derivaciones por la izquierda (o dos o más árboles de derivación).

- **Ej:**

Ambigüedad

- Strings con derivación "única"
  - Podemos definir funciones/propiedades sobre la derivación
  - Interpretabilidad

Def.

Un string w será *derivado ambiguamente* por G si tiene dos o más derivaciones por la izquierda (o dos o más árboles de derivación).

- **Ej:**
  - aaa por $S \to SS \mid a$.

Ambigüedad

- Strings con derivación "única"
  - Podemos definir funciones/propiedades sobre la derivación
  - Interpretabilidad

Def.

Un string w será *derivado ambiguamente* por G si tiene dos o más derivaciones por la izquierda (o dos o más árboles de derivación).

- **Ej:**
  - aaa por $S \to SS \mid a$.
  - $a + a \times a$ por $E \to E + E \mid E \times E \mid a$.

Ambigüedad

- Strings con derivación "única"
  - Podemos definir funciones/propiedades sobre la derivación
  - Interpretabilidad

Def.

Un string w será *derivado ambiguamente* por G si tiene dos o más derivaciones por la izquierda (o dos o más árboles de derivación).

- **Ej:**
  - aaa por $S \to SS \mid a$.
  - $a + a \times a$ por $E \to E + E \mid E \times E \mid a$.
  - $()()()$ por $S \to SS \mid (S) \mid \varepsilon$.

Ambigüedad

- Strings con derivación "única"
  - Podemos definir funciones/propiedades sobre la derivación
  - Interpretabilidad

Def.

Un string w será *derivado ambiguamente* por G si tiene dos o más derivaciones por la izquierda (o dos o más árboles de derivación).

- **Ej:**
  - aaa por $S \to SS \mid a$.
  - $a + a \times a$ por $E \to E + E \mid E \times E \mid a$.
  - $()()()$ por $S \to SS \mid (S) \mid \varepsilon$.

Def.

Una gramática será *ambigua* si deriva algún string ambiguamente.

Desambiguando gramáticas

- Nos gustaría diseñar gramáticas no-ambigüas

Desambiguando gramáticas

- Nos gustaría diseñar gramáticas no-ambigüas
- **Construcción usual:** dar roles distintos a la fuente de ambigüedad.

Desambiguando gramáticas

- Nos gustaría diseñar gramáticas no-ambigüas
- **Construcción usual:** dar roles distintos a la fuente de ambigüedad.
- **Ej:**

Desambiguando gramáticas

- Nos gustaría diseñar gramáticas no-ambigüas
- **Construcción usual:** dar roles distintos a la fuente de ambigüedad.
- **Ej:**
  - $S \to SS \mid a$.

Desambiguando gramáticas

- Nos gustaría diseñar gramáticas no-ambigüas
- **Construcción usual:** dar roles distintos a la fuente de ambigüedad.
- **Ej:**
  - $S \to SS \mid a.$
  - $E \to E + E \mid E \times E \mid a.$

Desambiguando gramáticas

- Nos gustaría diseñar gramáticas no-ambigüas
- **Construcción usual:** dar roles distintos a la fuente de ambigüedad.
- **Ej:**
  - $S \to SS \mid a.$
  - $E \to E + E \mid E \times E \mid a.$
  - $S \to SS \mid (S) \mid \varepsilon.$

Desambiguando gramáticas

- Nos gustaría diseñar gramáticas no-ambigüas
- **Construcción usual:** dar roles distintos a la fuente de ambigüedad.
- **Ej:**
  - $S \to SS \mid a.$
  - $E \to E + E \mid E \times E \mid a.$
  - $S \to SS \mid (S) \mid \varepsilon.$

- ¿Algoritmo?

Desambiguando gramáticas

- Nos gustaría diseñar gramáticas no-ambigüas
- **Construcción usual:** dar roles distintos a la fuente de ambigüedad.
- **Ej:**
  - $S \to SS \mid a.$
  - $E \to E + E \mid E \times E \mid a.$
  - $S \to SS \mid (S) \mid \varepsilon.$

- ¿Algoritmo?
- No siempre se puede: hay lenguajes inherentemente ambiguos.

Desambiguando gramáticas

- Nos gustaría diseñar gramáticas no-ambigüas
- **Construcción usual:** dar roles distintos a la fuente de ambigüedad.
- **Ej:**
  - $S \to SS \mid a.$
  - $E \to E + E \mid E \times E \mid a.$
  - $S \to SS \mid (S) \mid \varepsilon.$

- ¿Algoritmo?
- No siempre se puede: hay lenguajes inherentemente ambiguos.
- **Ej:**

Desambiguando gramáticas

- Nos gustaría diseñar gramáticas no-ambigüas
- **Construcción usual:** dar roles distintos a la fuente de ambigüedad.
- **Ej:**
  - $S \to SS \mid a.$
  - $E \to E + E \mid E \times E \mid a.$
  - $S \to SS \mid (S) \mid \varepsilon.$

- ¿Algoritmo?
- No siempre se puede: hay lenguajes inherentemente ambiguos.
- **Ej:**
  - $\{a^i b^j c^k : i = j \text{ o } j = k\}.$

Forma normal de Chomsky

Forma normal de Chomsky

- Para AP es útil asumir formas específicas

Forma normal de Chomsky

- Para AP es útil asumir formas específicas
- ¿Algo similar en GLC?

Forma normal de Chomsky

- Para AP es útil asumir formas específicas
- ¿Algo similar en GLC?
  - Útil al diseñar algoritmos sobre GLC

Forma normal de Chomsky

- Para AP es útil asumir formas específicas
- ¿Algo similar en GLC?
  - Útil al diseñar algoritmos sobre GLC
  - Diseño de parsers para GLC

Forma normal de Chomsky

- Para AP es útil asumir formas específicas
- ¿Algo similar en GLC?
  - Útil al diseñar algoritmos sobre GLC
  - Diseño de parsers para GLC
- Forma simplificada: Forma normal de Chomsky.

Forma normal de Chomsky

- Para AP es útil asumir formas específicas
- ¿Algo similar en GLC?
  - Útil al diseñar algoritmos sobre GLC
  - Diseño de parsers para GLC
- Forma simplificada: Forma normal de Chomsky.

Def.

Una GLC G está en *forma normal de Chomsky* si

Forma normal de Chomsky

- Para AP es útil asumir formas específicas
- ¿Algo similar en GLC?
  - Útil al diseñar algoritmos sobre GLC
  - Diseño de parsers para GLC
- Forma simplificada: Forma normal de Chomsky.

Def.

Una GLC G está en *forma normal de Chomsky* si

- La variable inicial no aparece al lado derecho de las reglas,

Forma normal de Chomsky

- Para AP es útil asumir formas específicas
- ¿Algo similar en GLC?
  - Útil al diseñar algoritmos sobre GLC
  - Diseño de parsers para GLC
- Forma simplificada: Forma normal de Chomsky.

Def.

Una GLC G está en *forma normal de Chomsky* si

- La variable inicial no aparece al lado derecho de las reglas,
- Las reglas son de la forma $A \to BC$ para variables A, B, C, o

Forma normal de Chomsky

- Para AP es útil asumir formas específicas
- ¿Algo similar en GLC?
  - Útil al diseñar algoritmos sobre GLC
  - Diseño de parsers para GLC
- Forma simplificada: Forma normal de Chomsky.

Def.

Una GLC G está en *forma normal de Chomsky* si

- La variable inicial no aparece al lado derecho de las reglas,
- Las reglas son de la forma $A \to BC$ para variables A, B, C, o
- De la forma $A \to a$ para variables A y terminales a, o

Forma normal de Chomsky

- Para AP es útil asumir formas específicas
- ¿Algo similar en GLC?
  - Útil al diseñar algoritmos sobre GLC
  - Diseño de parsers para GLC
- Forma simplificada: Forma normal de Chomsky.

Def.

Una GLC G está en *forma normal de Chomsky* si

- La variable inicial no aparece al lado derecho de las reglas,
- Las reglas son de la forma $A \to BC$ para variables A, B, C, o
- De la forma $A \to a$ para variables A y terminales a, o
- De la forma $S \to \varepsilon$ (sólo si $\varepsilon \in \mathcal{L}(G)$)

Forma normal de Chomsky

Teo.

Toda GLC tiene una GLC equivalente en forma normal de Chomsky.

Forma normal de Chomsky

Teo.

Toda GLC tiene una GLC equivalente en forma normal de Chomsky.

- **Idea:** corregir S al LD, $A \to \varepsilon$, $A \to B$, $A \to aBC \ldots$, $A \to ABC \ldots$

Forma normal de Chomsky

Teo.

Toda GLC tiene una GLC equivalente en forma normal de Chomsky.

- **Idea:** corregir S al LD, $A \to \varepsilon$, $A \to B$, $A \to aBC \ldots$, $A \to ABC \ldots$
  - S al LD: agregar nueva variable inicial.

Forma normal de Chomsky

Teo.

Toda GLC tiene una GLC equivalente en forma normal de Chomsky.

- **Idea:** corregir S al LD, $A \to \varepsilon$, $A \to B$, $A \to aBC \ldots$, $A \to ABC \ldots$
  - S al LD: agregar nueva variable inicial.
  - $A \to \varepsilon$: reemplazar dónde puede aparecer A por ε y eliminar.

Forma normal de Chomsky

Teo.

Toda GLC tiene una GLC equivalente en forma normal de Chomsky.

- **Idea:** corregir S al LD, $A \to \varepsilon$, $A \to B$, $A \to aBC \ldots$, $A \to ABC \ldots$
  - S al LD: agregar nueva variable inicial.
  - $A \to \varepsilon$: reemplazar dónde puede aparecer A por ε y eliminar.
  - $A \to B$: si $A \overset{*}{\Rightarrow} B$, podemos agregar las reglas de B a A.

Forma normal de Chomsky

Teo.

Toda GLC tiene una GLC equivalente en forma normal de Chomsky.

- **Idea:** corregir S al LD, $A \to \varepsilon$, $A \to B$, $A \to aBC\ldots$, $A \to ABC\ldots$
  - S al LD: agregar nueva variable inicial.
  - $A \to \varepsilon$: reemplazar dónde puede aparecer A por ε y eliminar.
  - $A \to B$: si $A \overset{*}{\Rightarrow} B$, podemos agregar las reglas de B a A.
  - $A \to aBC\ldots$: podemos agregar una nueva variable T_a, reemplazar la regla por $A \to T_aBC\ldots$ y agregar la regla $T_a \to a$.

Forma normal de Chomsky

Teo.

Toda GLC tiene una GLC equivalente en forma normal de Chomsky.

- **Idea:** corregir S al LD, $A \to \varepsilon$, $A \to B$, $A \to aBC\ldots$, $A \to ABC\ldots$
  - S al LD: agregar nueva variable inicial.
  - $A \to \varepsilon$: reemplazar dónde puede aparecer A por ε y eliminar.
  - $A \to B$: si $A \overset{*}{\Rightarrow} B$, podemos agregar las reglas de B a A.
  - $A \to aBC\ldots$: podemos agregar una nueva variable T_a, reemplazar la regla por $A \to T_aBC\ldots$ y agregar la regla $T_a \to a$.
  - $A \to ABC\ldots$: podemos separar en varias subreglas.

Forma normal de Chomsky

Teo.

Toda GLC tiene una GLC equivalente en forma normal de Chomsky.

- **Idea:** corregir S al LD, $A \to \varepsilon$, $A \to B$, $A \to aBC\ldots$, $A \to ABC\ldots$
  - S al LD: agregar nueva variable inicial.
  - $A \to \varepsilon$: reemplazar dónde puede aparecer A por ε y eliminar.
  - $A \to B$: si $A \overset{*}{\Rightarrow} B$, podemos agregar las reglas de B a A.
  - $A \to aBC\ldots$: podemos agregar una nueva variable T_a, reemplazar la regla por $A \to T_aBC\ldots$ y agregar la regla $T_a \to a$.
  - $A \to ABC\ldots$: podemos separar en varias subreglas.
- **Dem:** Inducción estructural.

Forma normal de Chomsky

Teo.

Toda GLC tiene una GLC equivalente en forma normal de Chomsky.

- **Idea:** corregir S al LD, $A \to \varepsilon$, $A \to B$, $A \to aBC\ldots$, $A \to ABC\ldots$
  - S al LD: agregar nueva variable inicial.
  - $A \to \varepsilon$: reemplazar dónde puede aparecer A por ε y eliminar.
  - $A \to B$: si $A \overset{*}{\Rightarrow} B$, podemos agregar las reglas de B a A.
  - $A \to aBC\ldots$: podemos agregar una nueva variable T_a, reemplazar la regla por $A \to T_aBC\ldots$ y agregar la regla $T_a \to a$.
  - $A \to ABC\ldots$: podemos separar en varias subreglas.
- **Dem:** Inducción estructural.
- **Proc:** Agregar nueva variable inicial, Eliminar ε, eliminar reglas unitarias, aislar terminales, binarizar reglas.

Forma normal de Chomsky

Teo.

Toda GLC tiene una GLC equivalente en forma normal de Chomsky.

- **Idea:** corregir S al LD, $A \to \varepsilon$, $A \to B$, $A \to aBC\ldots$, $A \to ABC\ldots$
  - S al LD: agregar nueva variable inicial.
  - $A \to \varepsilon$: reemplazar dónde puede aparecer A por ε y eliminar.
  - $A \to B$: si $A \overset{*}{\Rightarrow} B$, podemos agregar las reglas de B a A.
  - $A \to aBC\ldots$: podemos agregar una nueva variable T_a, reemplazar la regla por $A \to T_aBC\ldots$ y agregar la regla $T_a \to a$.
  - $A \to ABC\ldots$: podemos separar en varias subreglas.
- **Dem:** Inducción estructural.
- **Proc:** Agregar nueva variable inicial, Eliminar ε, eliminar reglas unitarias, aislar terminales, binarizar reglas.
- **Ej:**

Forma normal de Chomsky

Teo.

Toda GLC tiene una GLC equivalente en forma normal de Chomsky.

- **Idea:** corregir S al LD, $A \to \varepsilon$, $A \to B$, $A \to aBC\ldots$, $A \to ABC\ldots$
  - S al LD: agregar nueva variable inicial.
  - $A \to \varepsilon$: reemplazar dónde puede aparecer A por ε y eliminar.
  - $A \to B$: si $A \overset{*}{\Rightarrow} B$, podemos agregar las reglas de B a A.
  - $A \to aBC\ldots$: podemos agregar una nueva variable T_a, reemplazar la regla por $A \to T_aBC\ldots$ y agregar la regla $T_a \to a$.
  - $A \to ABC\ldots$: podemos separar en varias subreglas.
- **Dem:** Inducción estructural.
- **Proc:** Agregar nueva variable inicial, Eliminar ε, eliminar reglas unitarias, aislar terminales, binarizar reglas.
- **Ej:**
  - $S \to aSb \mid ab$

Forma normal de Chomsky

Teo.

Toda GLC tiene una GLC equivalente en forma normal de Chomsky.

- **Idea:** corregir S al LD, $A \to \varepsilon$, $A \to B$, $A \to aBC\ldots$, $A \to ABC\ldots$
  - S al LD: agregar nueva variable inicial.
  - $A \to \varepsilon$: reemplazar dónde puede aparecer A por ε y eliminar.
  - $A \to B$: si $A \overset{*}{\Rightarrow} B$, podemos agregar las reglas de B a A.
  - $A \to aBC\ldots$: podemos agregar una nueva variable T_a, reemplazar la regla por $A \to T_aBC\ldots$ y agregar la regla $T_a \to a$.
  - $A \to ABC\ldots$: podemos separar en varias subreglas.
- **Dem:** Inducción estructural.
- **Proc:** Agregar nueva variable inicial, Eliminar ε, eliminar reglas unitarias, aislar terminales, binarizar reglas.
- **Ej:**
  - $S \to aSb \mid ab$
  - $S \to aAB \mid B$, $A \to a \mid \varepsilon$, $B \to b$.

Dudas/Preguntas

Recapitulación sobre lenguajes libres de contexto

Lenguajes libres de contexto

- **Lenguaje libre de contexto:** aceptado por AP o generado por GLC.

Lenguajes libres de contexto

- **Lenguaje libre de contexto:** aceptado por AP o generado por GLC.
- Diseñar alguno para probar que un lenguaje es libre de contexto.

Lenguajes libres de contexto

- **Lenguaje libre de contexto:** aceptado por AP o generado por GLC.
- Diseñar alguno para probar que un lenguaje es libre de contexto.
- Cerrados bajo operaciones regulares: $\cup$, concatenación y $*$.

Lenguajes libres de contexto

- **Lenguaje libre de contexto:** aceptado por AP o generado por GLC.
- Diseñar alguno para probar que un lenguaje es libre de contexto.
- Cerrados bajo operaciones regulares: $\cup$, concatenación y $*$.
- GLC $\rightarrow$ AP.

Lenguajes libres de contexto

- **Lenguaje libre de contexto:** aceptado por AP o generado por GLC.
- Diseñar alguno para probar que un lenguaje es libre de contexto.
- Cerrados bajo operaciones regulares: $\cup$, concatenación y $*$.
- GLC $\rightarrow$ AP.
  - Adivinar reglas de producción

Lenguajes libres de contexto

- **Lenguaje libre de contexto:** aceptado por AP o generado por GLC.
- Diseñar alguno para probar que un lenguaje es libre de contexto.
- Cerrados bajo operaciones regulares: $\cup$, concatenación y $*$.
- GLC $\rightarrow$ AP.
  - Adivinar reglas de producción
  - Usar el stack para mantener la derivación por la izquierda

Lenguajes libres de contexto

- **Lenguaje libre de contexto:** aceptado por AP o generado por GLC.
- Diseñar alguno para probar que un lenguaje es libre de contexto.
- Cerrados bajo operaciones regulares: $\cup$, concatenación y $*$.
- GLC $\rightarrow$ AP.
  - Adivinar reglas de producción
  - Usar el stack para mantener la derivación por la izquierda
- AP $\rightarrow$ GLC.

Lenguajes libres de contexto

- **Lenguaje libre de contexto:** aceptado por AP o generado por GLC.
- Diseñar alguno para probar que un lenguaje es libre de contexto.
- Cerrados bajo operaciones regulares: $\cup$, concatenación y $*$.
- GLC $\rightarrow$ AP.
  - Adivinar reglas de producción
  - Usar el stack para mantener la derivación por la izquierda
- AP $\rightarrow$ GLC.
  - Simplificación: AP con único estado final, sin `replace` y acepta con pila vacía.

Lenguajes libres de contexto

- **Lenguaje libre de contexto:** aceptado por AP o generado por GLC.
- Diseñar alguno para probar que un lenguaje es libre de contexto.
- Cerrados bajo operaciones regulares: $\cup$, concatenación y $*$.
- GLC $\rightarrow$ AP.
  - Adivinar reglas de producción
  - Usar el stack para mantener la derivación por la izquierda
- AP $\rightarrow$ GLC.
  - Simplificación: AP con único estado final, sin `replace` y acepta con pila vacía.
  - **PD:** sobre A_{pq}="ir de p a q con pila vacía"

Lenguajes libres de contexto

- **Lenguaje libre de contexto:** aceptado por AP o generado por GLC.
- Diseñar alguno para probar que un lenguaje es libre de contexto.
- Cerrados bajo operaciones regulares: $\cup$, concatenación y $*$.
- GLC $\rightarrow$ AP.
  - Adivinar reglas de producción
  - Usar el stack para mantener la derivación por la izquierda
- AP $\rightarrow$ GLC.
  - Simplificación: AP con único estado final, sin `replace` y acepta con pila vacía.
  - **PD:** sobre A_{pq}="ir de p a q con pila vacía"
  - Para todo r intermedio, agregar $A_{pq} \rightarrow A_{pr}A_{rq}$

Lenguajes libres de contexto

- **Lenguaje libre de contexto:** aceptado por AP o generado por GLC.
- Diseñar alguno para probar que un lenguaje es libre de contexto.
- Cerrados bajo operaciones regulares: $\cup$, concatenación y $*$.
- GLC $\rightarrow$ AP.
  - Adivinar reglas de producción
  - Usar el stack para mantener la derivación por la izquierda
- AP $\rightarrow$ GLC.
  - Simplificación: AP con único estado final, sin `replace` y acepta con pila vacía.
  - **PD:** sobre A_{pq}="ir de p a q con pila vacía"
  - Para todo r intermedio, agregar $A_{pq} \rightarrow A_{pr}A_{rq}$
  - Apilamos y desapilamos el mismo símbolo: $A_{pq} \rightarrow aA_{rs}b$

Lenguajes libres de contexto

- **Lenguaje libre de contexto:** aceptado por AP o generado por GLC.
- Diseñar alguno para probar que un lenguaje es libre de contexto.
- Cerrados bajo operaciones regulares: $\cup$, concatenación y $*$.
- GLC $\rightarrow$ AP.
  - Adivinar reglas de producción
  - Usar el stack para mantener la derivación por la izquierda
- AP $\rightarrow$ GLC.
  - Simplificación: AP con único estado final, sin `replace` y acepta con pila vacía.
  - **PD:** sobre A_{pq}="ir de p a q con pila vacía"
  - Para todo r intermedio, agregar $A_{pq} \rightarrow A_{pr}A_{rq}$
  - Apilamos y desapilamos el mismo símbolo: $A_{pq} \rightarrow aA_{rs}b$
  - $A_{pp} \rightarrow \varepsilon$

Lenguajes libres de contexto

- **Lenguaje libre de contexto:** aceptado por AP o generado por GLC.
- Diseñar alguno para probar que un lenguaje es libre de contexto.
- Cerrados bajo operaciones regulares: $\cup$, concatenación y $*$.
- GLC $\rightarrow$ AP.
  - Adivinar reglas de producción
  - Usar el stack para mantener la derivación por la izquierda
- AP $\rightarrow$ GLC.
  - Simplificación: AP con único estado final, sin `replace` y acepta con pila vacía.
  - **PD:** sobre A_{pq}="ir de p a q con pila vacía"
  - Para todo r intermedio, agregar $A_{pq} \rightarrow A_{pr}A_{rq}$
  - Apilamos y desapilamos el mismo símbolo: $A_{pq} \rightarrow aA_{rs}b$
  - $A_{pp} \rightarrow \varepsilon$
- **Tópicos en GLC:** Ambigüedad y forma normal de Chomsky.

Wrap-Up

- **Próx. clase:**

Wrap-Up

- **Próx. clase:**
  - Lenguajes no libres de contexto