Generación de Lenguaje Natural y Aplicaciones

Carlos Areces y Luciana Benotti

{carlos.areces,luciana.benotti}@gmail.com

INRIA Nancy Grand Est, Nancy, France
Universidad Nacional de Córdoba, Córdoba, Argentina

ELiC 2010 - Buenos Aires - Argentina

Lo que Vemos Hoy

- Las Tareas Básicas de GLN
- Tree Adjoining Grammars (TAGS)
- TAGs for Natural Language Processing

Lo que Veremos Hoy

- Las Tareas Básicas de GLN
  - Document Planning
  - Microplanning
  - Surface Realization
- Tree Adjoining Grammars (TAGS)
- TAGs for Natural Language Processing

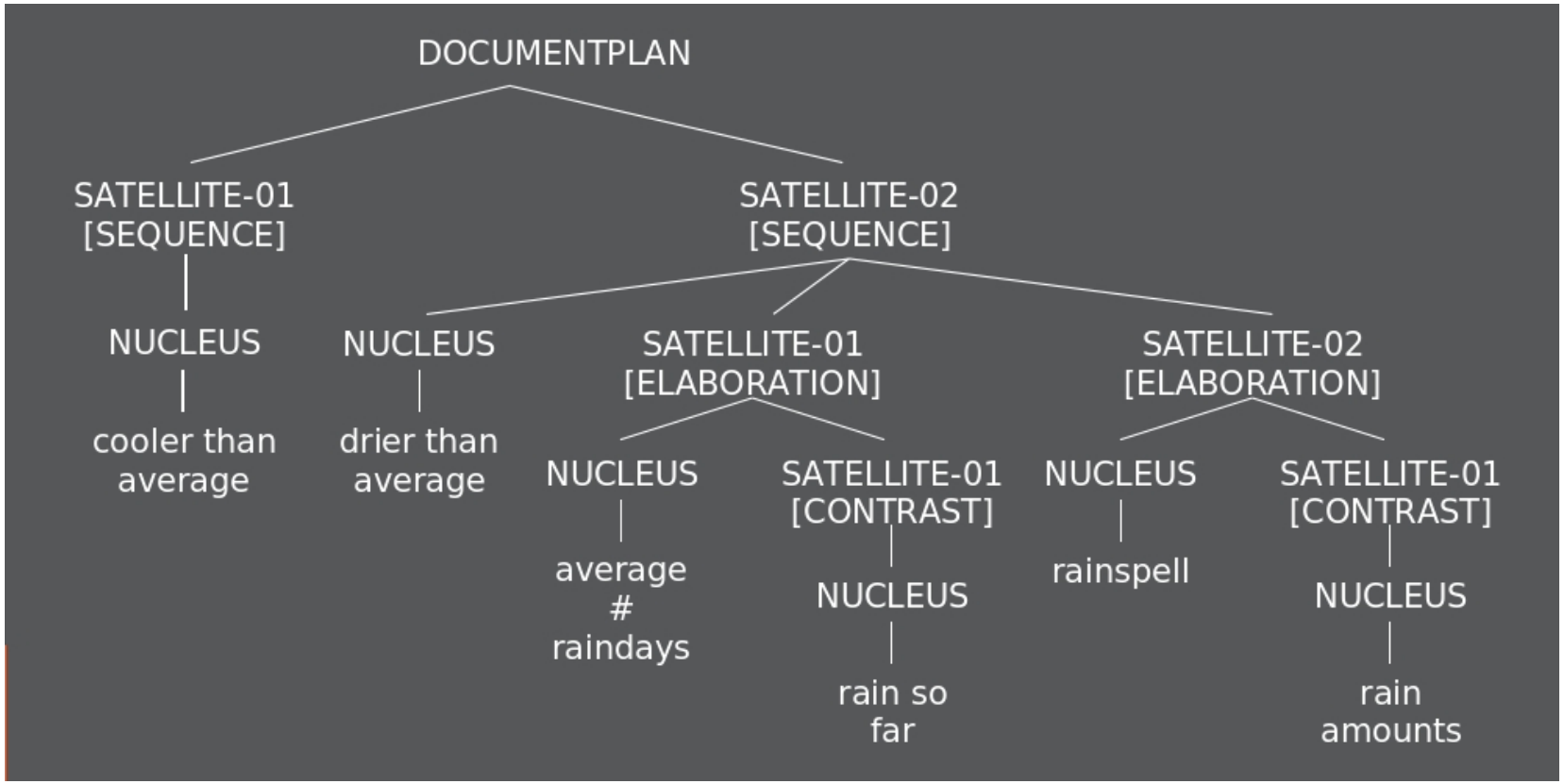
Las Tareas Básicas en un sistema de GLN

Content Determination Document Structuring	Document planning
Aggregation Lexicalisation Referring Expression Generation	Micro-planning
Linguistic Realisation Structure Realisation	Surface realization

Y si nos Salteamos Microplanning?

- Podríamos saltearnos completamente la etapa de microplanning
- Producimos una sentencia por cada mensaje del plan de documento
  - Para cada tipo de mensaje en un nodo del plan decidimos, individualmente, como será realizado

Ejemplo de un Document Plan



Un Realizador Simple

- Para el mensaje MonthlyTemperatureMsg:
  TempString = case (TEMP - AVERAGETEMP)
  (2.0 – 2.9): 'very much warmer than average.'
  (1.0 – 1.9): 'much warmer than average.'
  (0.1 – 0.9): 'slightly warmer than average.'
  (-0.1 – -0.9): 'slightly cooler than average.'
  (-1.0 – -1.9): 'much cooler than average.'
  (-2.0 – -2.9): 'very much cooler than average.'
- Sentencia = 'The month was' + TempString

Un Mensaje por Sentencia

- El resultado sería:

  The month was cooler than average.
  The month was drier than average.
  There were the average number of rain days.
  The total rain for the year so far is well below average.
  There was rain on every day for 8 days from 11th to 18th.
  Rainfall amounts were mostly small.

Un Mensaje por Sentencia

- El resultado sería:

  The month was cooler than average.
  The month was drier than average.
  There were the average number of rain days.
  The total rain for the year so far is well below average.
  There was rain on every day for 8 days from 11th to 18th.
  Rainfall amounts were mostly small.

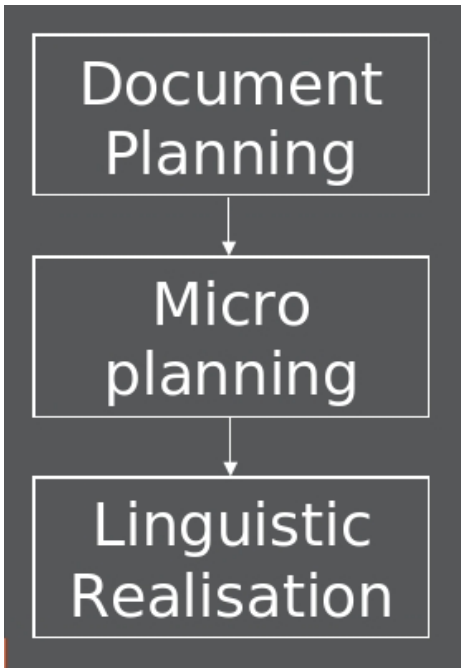
- El texto target:

  The month was cooler and drier than average, with the average number of rain days, but the total rain for the year so far is well below average. Although there was rain on every day for 8 days from 11th to 18th, rainfall amounts were mostly small.

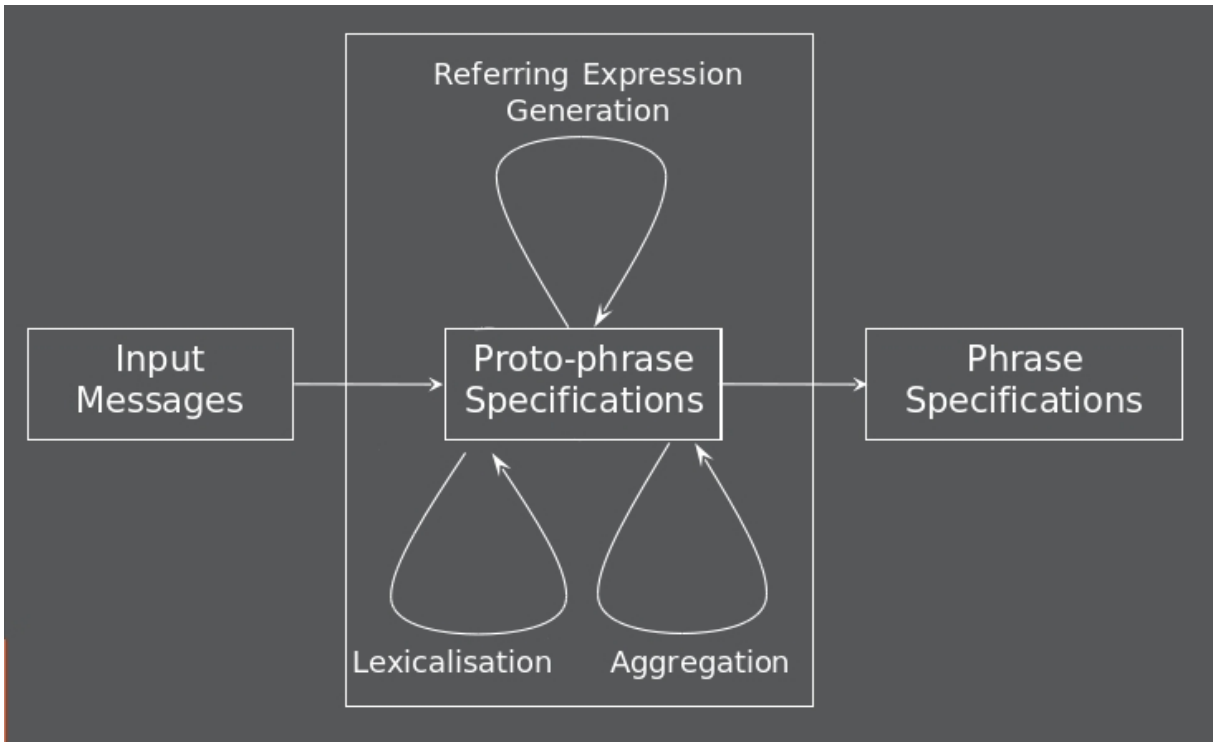
Microplanning

- Objetivo:
  - Convertir el document plan en una secuencia de sentencias o especificación de frases.
- Tareas:
  - Agregación de párrafos y sentencias
  - Lexicalización
  - Referencia

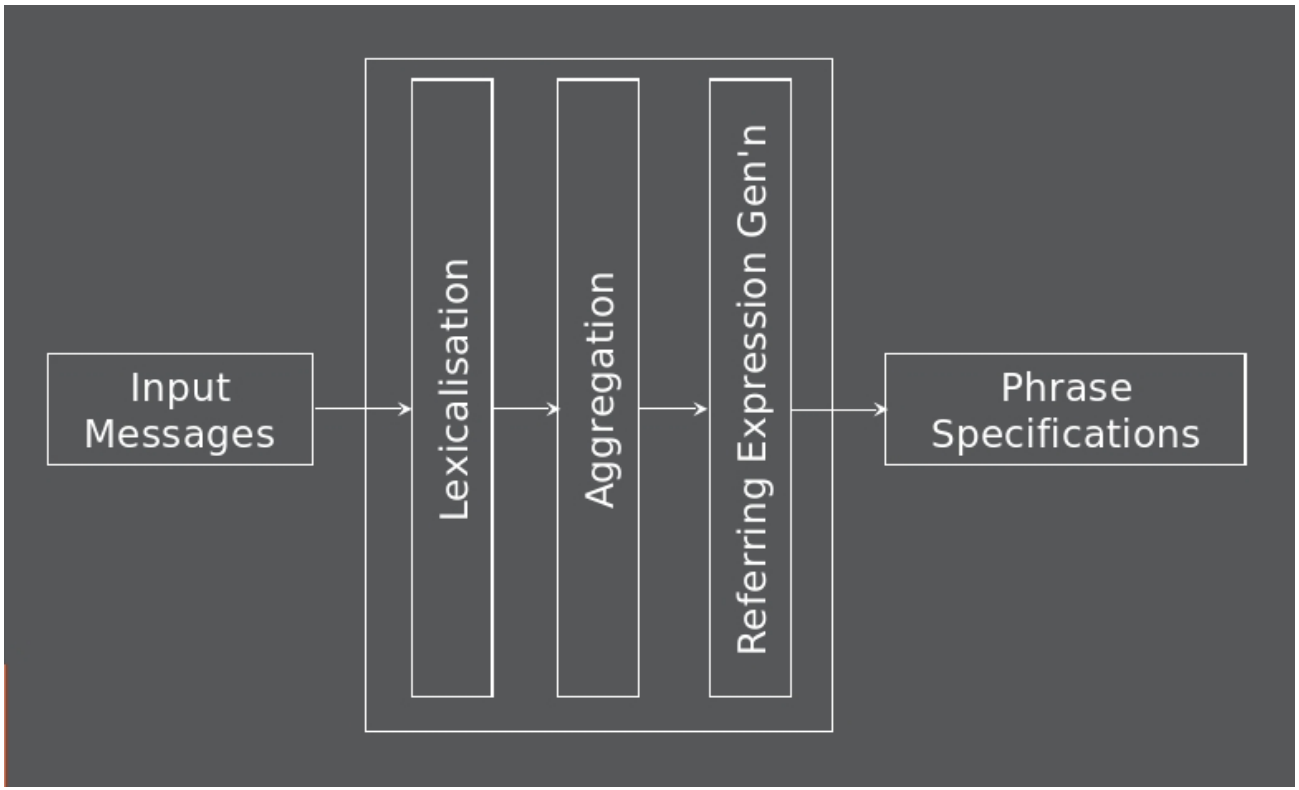
La Arquitectura



Interacciones en Microplanning



Microplanning en un Pipeline



Agregación

- Podemos combinar (agrupar) texto de acuerdo a
  - el contenido de información
  - posibilidades de realización
- Algunas Posibilidades:
  - Conjunción
  - Elipsis
  - Embedding
  - Introducción mediante Conjuntos

Ejemplos

- Sin Agregación:
  - Heavy rain fell on the 27th.
    Heavy rain fell on the 28th.
- Agregación via conjunción simple:
  - Heavy rain fell on the 27th and heavy rain fell on the 28th.
- Agregación via elipsis:
  - Heavy rain fell on the 27th and [] on the 28th.
- Agregación via introducción de conjuntos:
  - Heavy rain fell on [the 27th and 28th].

Ejemplos: Embedding

- Sin agregación:
  - March had a rainfall of 120mm.
    It was the wettest month.
- Con agregación:
  - March, which was the wettest month, had a rainfall of 120mm.

Heurísticas de Elección

Usualmente, hay muchas formas en que un conjunto de mensajes pueden agregarse: cómo elegimos?

- reglas convencionales y de estilo
- respetar propiedades estructurales
  - Por ejemplo, agregar solamente mensajes que son hermanos en el árbol del plan de documento
- tomando en cuenta objetivos pragmáticos

Objetivos Pragmáticos: STOP

- Hacer el texto más amigable:

  It's clear from your answers that you don't feel too happy about being a smoker and it's excellent that you are going to try to stop.

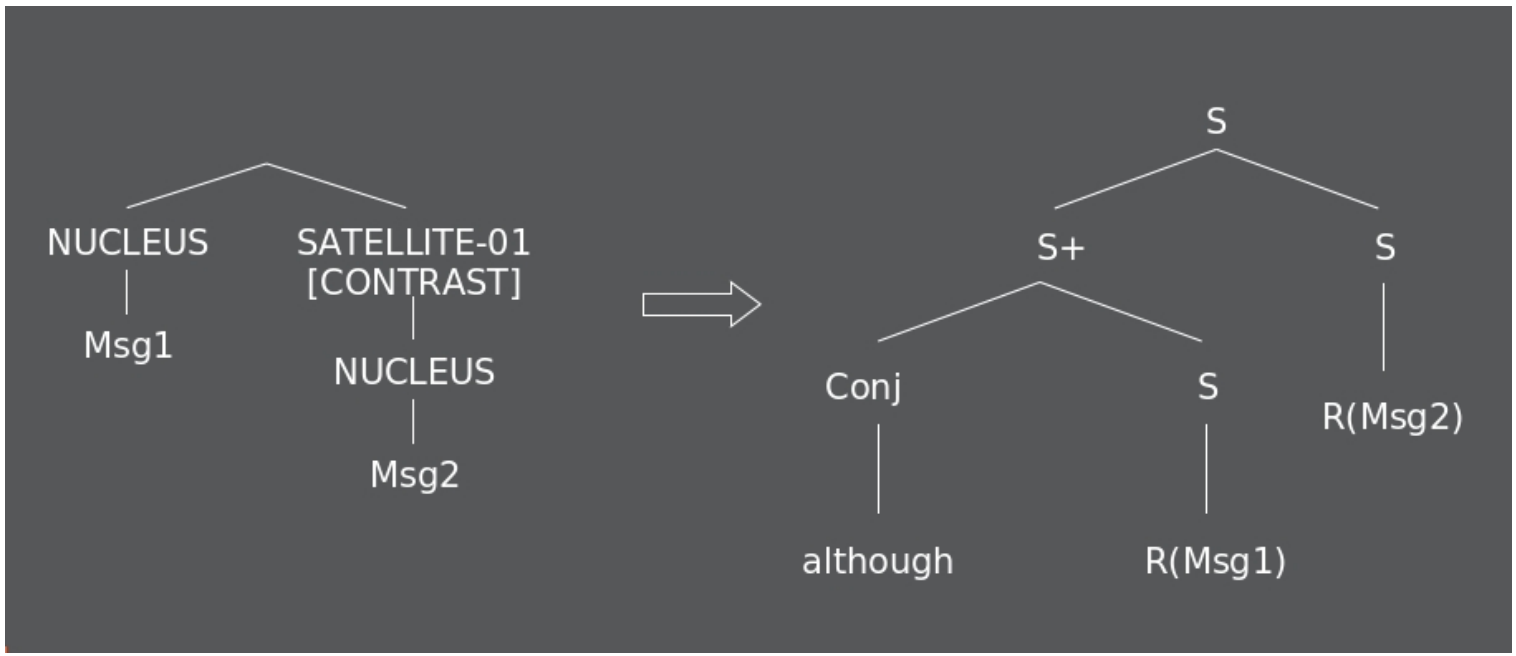
- Hacer el texto más simple (para personas con dificultades de lectura):

  It's clear from your answers that you don't feel too happy about being a smoker. It's excellent that you are going to try to stop.

Agregación en WeatherReporter

- Teniendo en cuenta las relaciones retóricas:
  - Si dos mensajes están en la relación SEQUENCE, pueden ser agrupados al mismo nivel.
  - Si un mensaje es una ELABORATION de otro, puede ser agrupado al mismo nivel, o embebido como una clausula o frase menor.
  - Si un mensaje es un CONTRAST de otro, puede ser agrupado al mismo nivel, o embebido como una clausula o frase menor.

Ejemplo de Regla de Agregación



Lexicalización

- Es el proceso de elección de palabras para comunicar la información de un mensaje
- Métodos:
  - templates
  - árboles de decisión
  - algoritmos de reescritura de grafos

Elección Léxica

Si múltiples lexicalizaciones son posibles, considerar:

- el conocimiento del usuario y sus preferencias
- consistencia con usos anteriores
  - aunque en algunos casos, puede ser posible variar lexemas
- interacción con otros aspectos de microplanning
- objetivos pragmáticos
  - It is encouraging that you have many reasons to stop.
    (significado mas preciso)
  - It's good that you have a lot of reasons to stop.
    (más fácil de leer)

WeatherReporter: Variaciones al describir MonthlyRainfallMsg

- Variaciones en la categoría sintáctica:
  S: [rainfall was very poor]
  NP: [a much worse than average rainfall]
  AP: [much drier than average]
- Variaciones semánticas:
  Valores absolutos:
    [very dry]
    [a very poor rainfall]
  Valores comparativos:
    [a much worse than average rainfall]
    [much drier than average]

WeatherReporter: Agregación y Lexicalización

Muchos resultados diferentes son posibles

The month was cooler and drier than average. There were the average number of rain days, but the total rain for the year so far is well below average. There was rain on every day for 8 days from 11th to 18th, but rainfall amounts were mostly small.

The month was cooler and drier than average. Although the total rain for the year so far is well below average, there were the average number of rain days. There was a small mount of rain on every day for 8 days from 11th to 18th.

Generación de Expresiones Referenciales

Como identificamos objetos y entidades específicas del dominio?

- Dos temas que se discuten clásicamente:
  - Primera referencia a un objeto
  - Referencias subsequentes a un objeto previamente introducido (y 'notorio')

Referencia Inicial

Introduciendo un objeto en el discurso

- usar un nombre completo
  - Jeremy
- relacionar con un objeto que ya fue introducido (y es notorio)
  - Jane's goldfish
- especificar ubicación física
  - the goldfish in the bowl on the table

Un tema todavía no resuelto, y en investigación.

Referencias Subsequentes

Algunas posibilidades

- Pronombres
  - It swims in circles.
- Frases nominales definidas (definite NPs)
  - The goldfish swims in circles.
- Nombres propios (quizas abreviados)
  - Jeremy swims in circles.

Eligiendo el Modo de Referencia

Algunas sugerencias en la literatura:

- usar un pronombre si este refiere a una entidad mencionada en la frase anterior, siempre y cuando no hay otra entidad en la frase anterior que pueda ser referica con el mismo pronombre
- si no, usar un nombre, si este existe
- si no, generar un definite NP

También es importante seguir las convenciones del género – por ejemplo, se usan más pronombres en artículos periodísticos que en manuales técnicos

Ejemplo

I am taking the Caledonian Express tomorrow. It is a much better train than the Grampian Express. The Caledonian has a real restaurant car, while the Grampian just has a snack bar. The restaurant car serves wonderful fish, while the snack bar serves microwaved mush.

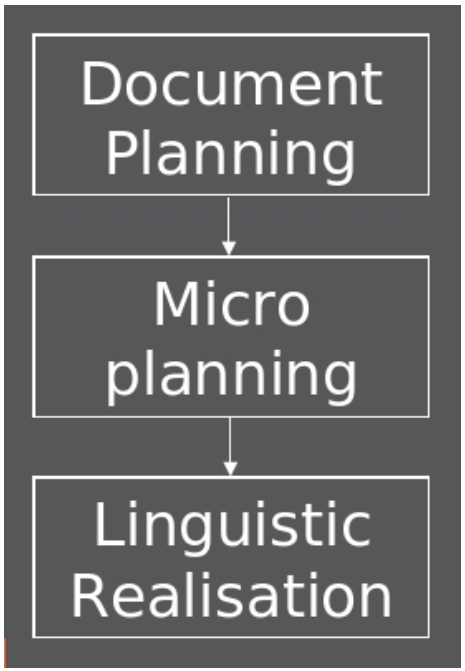
Generación de Expresiones Referenciales en WeatherReporter

- Referencias a meses:
  - June 1999
  - June
  - the month
  - next June
- Relativamente simple, puede harcodearse durante la etapa de document planning

Temas de Investigación

- Cómo guiar las elecciones durante microplanning?
- Cómo hacer agregación de alto nivel, i.e., como formar parrafos a partir de sentencias?
- Cómo guiar las elecciones de lexicalización, en particular, como lexicalizar cuando conceptos del dominio no pueden traducirse directamente y en forma fácil en palabras?
- Cuál es la mejor manera de referirnos a un objeto (en particular en una primera referencia)?

La Arquitectura



Realización

- Objetivo:
  - Convertir las especificaciones de texto en texto en lenguaje natural
- Motivo:
  - intentar separar lo más posible las particularidades de un lenguaje dado, del resto del sistema de GLN

Tareas de Realización

- Realización de Estructura:
  - Elegir el markup necesario para transmitir la estructura del documento
- Realización Lingüística:
  - Insertar palabras funcionales (functional words). E.g., artículos, preposiciones, etc.
  - Elegir la inflexión correcta de las palabras de contenido (content words). E.g., sustantivos, adjetivos, etc.
  - Ordenar las palabras dentro de una frase
  - Aplicar reglas ortográficas y morfológicas (inflexion de palabras, reglas de puntuación, etc.)

Realización de Estructura

- Agregar markup al documento:
- Ejemplo: formas de marcar el fin de párrafo

HTML	<P>
LaTeX	(blank line)
RTF	\par
SABLE (speech)	<BREAK>

- Depende del formato en que se representará el documento
- Usualmente puede resolverse con reglas simples de mapeo

Realización Lingüística

Diferentes técnicas:

- Especificación de gramáticas bidireccionales
- Especificación de gramáticas específicas para generación
- Mecanismos basados en templates

Especificación de Gramáticas Bidireccionales

- Idea Principal: Utilización de una misma gramática para realización y parsing
- Puede ser expresada como un conjunto de mapeos declarativos entre estructuras sintácticas y semánticas
- Lo que diferirá (generación vs. interpretación) serán los algoritmos que utilicen la gramática
- Elegantes en teoría
- Hoy en dá, son utilizadas en algunos sistemas de traducción automática, pero casi nunca en sistemas de GLN aplicada.
  - Dificultad de construir gramáticas bidireccionales de gran tamaño
  - Dificultad de generar frases específicas con estas gramáticas

Gramáticas Especialmente Definidas para Generación

- La gramática provee un árbol de decisión para la realización
- Las elecciones se hacen en base a la especificación en el plan del documento
- Estas gramáticas se pueden utilizar sólo para generación
- Existe software y estandares para construir estas gramáticas

Observación

- Los dos enfoques anteriores apuntan a generar texto de amplia cobertura lingüística y de carácter sofisticado.
- La mayoría de las aplicaciones no require este nivel de sofisticación: en muchos casos los objetivos de generación se pueden obtener via templates o 'canned text'
- Lo que si puede suceder es que ciertas partes del problema de generación requiera más sofisticación que otras.
- En muchos casos la solución es combinar canned text, templates y 'realización en serio'

Temas de Investigación

- Cómo combinamos distintas técnicas y formalismos lingüísticos?
- Cuáles son los costos y beneficions de usar templates en vez de 'realización en serio'?
- En qué casos hay interacción entre realización de estructura y realización lingüística?

Lo que Vemos Hoy

- Las Tareas Básicas de GLN
- Tree Adjoining Grammars (TAG)
- TAGs for Natural Language Processing

Lo que Vemos Hoy

- Las Tareas Básicas de GLN
- Tree Adjoining Grammars (TAG)
  - Capacidad Generativa Débil vs. Fuerte
  - Introduciendo TAG
  - Propiedades de TAG, Complejidad
  - Lexicalización de Gramáticas
- TAGs y Surface Realization

Gramáticas Formales

- Podemos usar gramáticas formales para generar lenguages.
- Por ejemplo la gramática

$$S \quad \rightarrow \quad aSb \mid ab$$

  genera $a^n b^n$ para $n \geq 1$
- Podemos usar este tipo de gramáticas para generar lenguaje natural?

John loves Mary

$$
\begin{aligned}
\text{S} &\rightarrow \text{NP VP} \\
\text{VP} &\rightarrow \text{V NP} \mid \text{VP ADV} \\
\text{NP} &\rightarrow \text{John} \mid \text{Mary} \\
\text{V} &\rightarrow \text{loves} \\
\text{ADV} &\rightarrow \text{passionately}
\end{aligned}
$$

$L(G) = \{\text{John loves Mary, John loves Mary passionately,}\ldots\}$

Gramáticas Formales

- Recordemos de todas formas que la tarea de GLN es
  - Transformar información no lingüística
  - en texto comprensible en lenguaje natural
- Podemos usar este tipo de gramática para generar a partir de información no lingüística?
- Que gramática tenemos que usar?

La Jerarquía de Chomsky

Una gramática G es una tupla (N, T, R, S) donde

- N es un conjunto de símbolos (No Terminales)
- T es el conjunto de símbolos disjunto de N (Terminales)
- R es una relacion en $(N \cup T)^* \times (N \cup T)^*$
- S es un símbolo de N.

La Jerarquía de Chomsky

Una gramática G es una tupla (N, T, R, S) donde

- N es un conjunto de símbolos (No Terminales)
- T es el conjunto de símbolos disjunto de N (Terminales)
- R es una relacion en $(N \cup T)^* \times (N \cup T)^*$
- S es un símbolo de N.

Sea $G = (N, T, R, S)$ una gramática donde α, β, $\gamma \in (N \cup T)^*$

- unrestricted o type-0 grammars: $\alpha \rightarrow \beta$, tal que $\alpha \neq \epsilon$
- context-sensitive grammars: $\alpha A \beta \rightarrow \alpha \gamma \beta$, tal que $\gamma \neq \epsilon$
- context-free grammars: $A \rightarrow \gamma$
- regular grammars: A $\rightarrow$ a B or A $\rightarrow$ a

Qué Gramática Tenemos que Usar

- Olvidemonos por un momento del problema de generación.
- Supongamos que queremos escribir una gramática que contenga un lenguaje natural como el Ingles o el Castellano
- Es decir una gramática G tal que $L(G)$ sea un subconjunto interesante de un lenguaje natural
- Que tipo de gramática tiene que ser G? E.g., puede ser una gramática regular?

Capacidad Generativa Débil vs. Fuerte

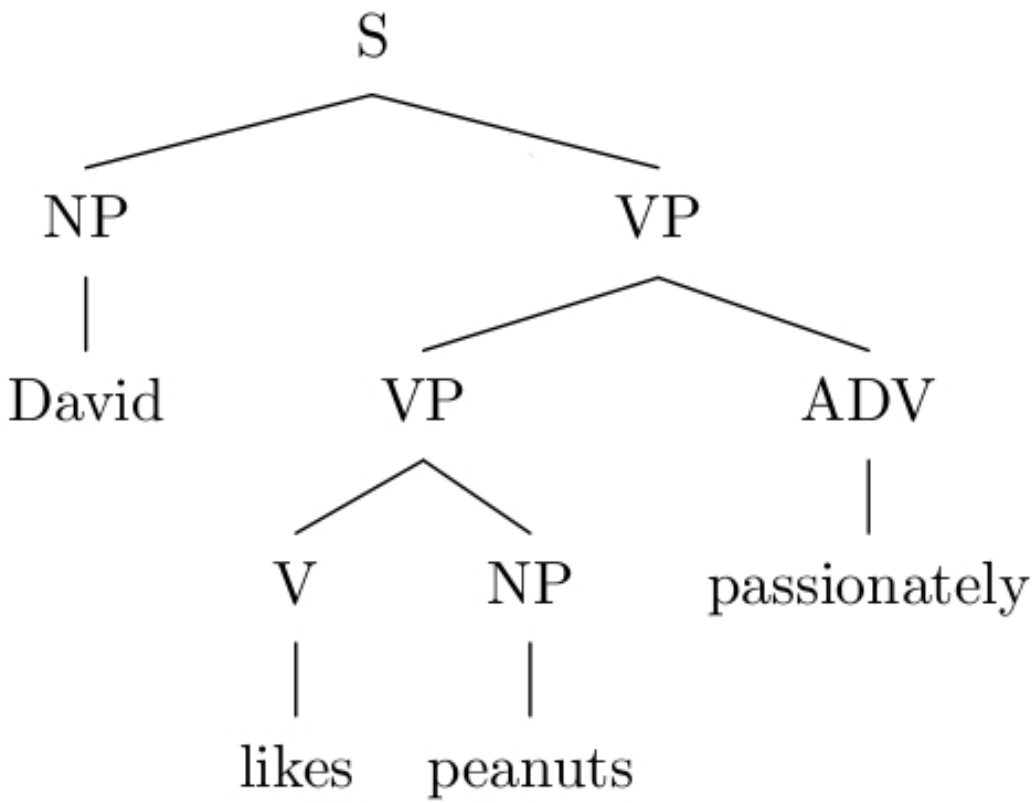
- Que queremos generar? $L(G)$ o $T(G)$
- La capacidad generativa débil $L(G)$ de una gramática es el conjunto de strings (o lenguaje) que la gramática genera/reconoce, e.g. 0^n1^n para $n \geq 0$
- La capacidad generativa fuerte $T(G)$ de una gramática es el conjunto de derivaciones (usualmente un conjunto de árboles) de una gramática

Capacidad Generativa Débil vs. Fuerte

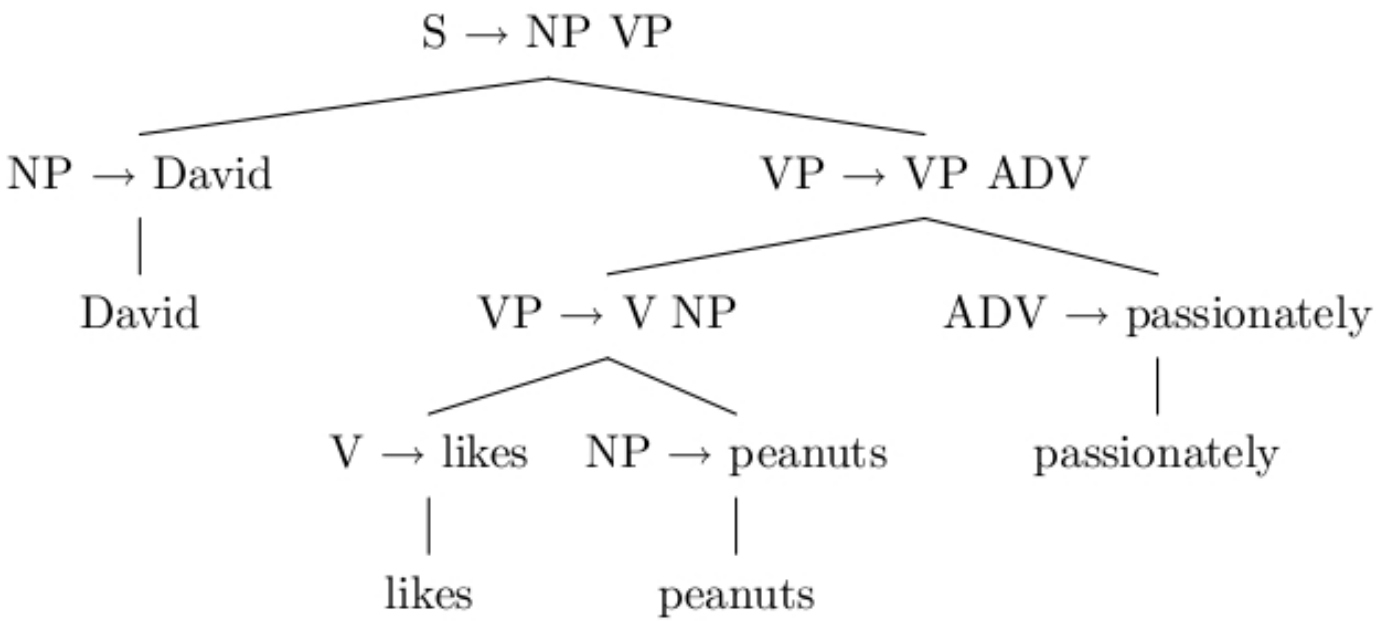
$$
\begin{aligned}
S &\rightarrow NP\ VP \\
VP &\rightarrow V\ NP \mid VP\ ADV \\
NP &\rightarrow John \mid Mary \mid David \mid peanuts \\
V &\rightarrow loves \mid likes \\
ADV &\rightarrow passionately
\end{aligned}
$$

$$L(G) = \{\text{David likes peanuts, David likes peanuts passionately,}\ldots\}$$

Arbol Derivado / Arbol de Parsing



No Confundir con: Arbol de Derivación

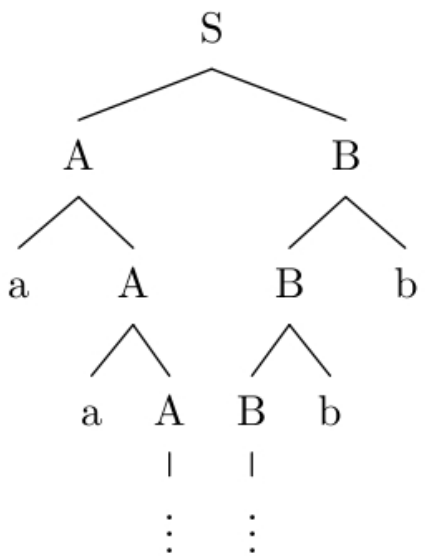


Mismo $L(G)$, Distinto $T(G)$

Dos gramáticas pueden tener el mismo $L(G)$ pero distingo $T(G)$

$$
\begin{array}{lcl}
S & \rightarrow & AB \\
A & \rightarrow & aA \mid a \\
B & \rightarrow & Bb \mid b
\end{array}
$$

$L(G) = a^+ b^+$ $\qquad$ $T(G) =$



Lenguaje Natural y Complejidad Descriptiva

- La pregunta que nos estamos haciendo entonces es: existe algun tipo de lenguaje formal que describa (al menos ciertos aspectos de un fragmento de) un lenguaje natural?
- Por ejemplo, supongamos que podemos abstraer algun fenómeno lingüístico de tal forma que lo podemos representar como el lenguaje

$$\{ww^R \mid \text{ donde } w \in \{a, b\}^*, w^R \text{ es el reverso de } w\}$$

- Podemos representar este lenguaje con una gramática regular? O necesitamos subir en la jerarquía?

Más Allá de Libre de Contexto

- Actualmente, se considera que ciertos aspectos de muchos lenguajes naturales no pueden ser modelados mediante gramáticas libres de contexto
- Si consideramos la capacidad generativa fuerte (i.e., $T(G)$) mostrar este resultado es relativamente simple.
  - Los lenguajes libres de contexto no pueden generar lenguajes con 'dependencias cruzadas.' E.g., $a^n b^m c^n d^m$
  - Es relativamente fácil dar ejemplos de $T(G)$ correspondientes a fenómenos lingüísticos con dependencias cruzadas.

Más Allá de Libre de Contexto

- Pero un resultado sobre capacidad generativa fuerte necesariamente está asociado a una determinada teoría lingüística (e.g., analisará sentencias de una forma determinada, usando ciertas categorías gramaticales)
- No sabemos qué teoría lingüística es la correcta!
- Lo único que conocemos son las sentencias que producimos, el resto está solo en nuestras cabezas.
- Por eso, un resultado más fuerte (y más interesante) sería poder demostrar que cierto lenguaje natural no es libre de contexto a nivel de capacidad generativa débil.

Más Allá de Libre de Contexto en $L(G)$

- Pero demostrar esto es difícil
- Consideremos $L_1 = \{a^n b^n\}$ es libre de contexto mientras que $L_2 = \{a^* b^*\}$ es regular y $L_1 \subset L_2$.
- Es decir, podriamos 'hacer trampa' y elegir algun subset del lenguaje natural particularmente difícil para demostrar nuestro resultado, mientras que el lenguaje completo tiene menor complejidad.

El Lenguaje Humano no es un Lenguaje Libre de Contexto

- Existen en la literatura diferentes argumentos e intentos de demostraciíon de que la capacidad generativa débil del lenguaje humano está mas alla de las gramáticas libres de contexto.
- (Pullum, 1982) es un compedio de muchos de estos argumentos y de sus problemas.
- (Shieber, 1985) y (Huybregts, 1984) proveyeron finalmente un argumento fuerte utilizando ejemplos de Alemán-Suizo.

Ejemplo: Más alla de Lenguajes Regulares

- Consideremos 'center embedding' en Inglés

the shares that the broker recommended were bought

$$\Rightarrow$$

$$N_1 N_2 V_2 V_1$$

the moment when the shares that the broker recommended were bought has passed

$$\Rightarrow$$

$$N_1 N_2 N_3 V_3 V_2 V_1$$

Ejemplo: Más alla de Lenguajes Regulares

- Consideremos 'center embedding' en Inglés

  the shares that the broker recommended were bought

  $\Rightarrow$

  $N_1 N_2 V_2 V_1$

  the moment when the shares that the broker recommended were bought has passed

  $\Rightarrow$

  $N_1 N_2 N_3 V_3 V_2 V_1$

- Podemos modelar este tipo de frases con una gramática como

  $$S \rightarrow aSb$$

  que produce dependencias anidadas como en $a_1 a_2 a_3 b_3 b_2 b_1$

Ejemplo: Más alla de Lenguajes Regulares

- Consideremos 'center embedding' en Inglés

  the shares that the broker recommended were bought

  $\Rightarrow$

  $N_1 N_2 V_2 V_1$

  the moment when the shares that the broker recommended were bought has passed

  $\Rightarrow$

  $N_1 N_2 N_3 V_3 V_2 V_1$

- Podemos modelar este tipo de frases con una gramática como

  $$S \rightarrow aSb$$

  que produce dependencias anidadas como en $a_1 a_2 a_3 b_3 b_2 b_1$
- Cuántas veces podemos anidar de esta forma? Por ejemplo, podemos construir una frase que suene natural y que incluya cuatro verbos?

Competencia vs. Performance

- Que pasa si no podemos embeber más de 3 o 4 sentencias?
- Si existe un limite finito (no importa que grande sea), entonce si podemos modelar este fragmento usando un lenguaje regular!
- Usualmente vamos a querer diferenciar competencia de performance. I.e., lo que consideramos gramaticalmente correcto, de lo que realmente producimos (dadas, por ejemplo, nuestras limitaciones de memoria).
- En el caso anterior diríamos que la performance puede modelarse con un lenguaje regular, mientras que la competencia necesita un lenguaje libre de contexto.

Ejemplo: Mas allá de Libre de Contexto

- Varios lenguajes tienen dependencias cruzadas: Holandes (Bresnan et al., 1982), Alemán-Suizo (Shieber, 1984), Tagalog (Rambow and MacLachlan, 2002)

Ejemplo: Mas allá de Libre de Contexto

- Varios lenguajes tienen dependencias cruzadas: Holandes (Bresnan et al., 1982), Alemán-Suizo (Shieber, 1984), Tagalog (Rambow and MacLachlan, 2002)
- Alemán-Suizo:

...	mer	em Hans	es huss	halfed	aastriiche
...	we	Hans	the house	helped	paint
		N_1	N_2	V_1	V_2

we helped Hans paint the house

Lo que Vemos Hoy

- Las Tareas Básicas de GLN
- Tree Adjoining Grammars (TAG)
- TAGs y Surface Realization

Capacidad Generativa Débil vs. Fuerte
Introduciendo TAG
Propiedades de TAG, Complejidad
Lexicalización de Gramáticas

Gramáticas de Arboles

- En la discusion anterior usamos ambos $L(G)$ and $T(G)$ para discutir distintos aspectos del lenguaje humano.
- Como dijimos, sabemos lo que $L(G)$ es y solo podemos teorizar hacerca de lo que $T(G)$ es.

Gramáticas de Arboles

- En la discusion anterior usamos ambos $L(G)$ and $T(G)$ para discutir distintos aspectos del lenguaje humano.
- Como dijimos, sabemos lo que $L(G)$ es y solo podemos teorizar hacerca de lo que $T(G)$ es.
- Pero una vez que fijamos una teoría lingüística, intuitivamente $T(G)$ es tan important o más que $L(G)$.
- De alguna forma, podemos decir que $L(G)$ es solo el resultado de las características de $T(G)$.

Gramáticas de Arboles

- En la discusion anterior usamos ambos $L(G)$ and $T(G)$ para discutir distintos aspectos del lenguaje humano.
- Como dijimos, sabemos lo que $L(G)$ es y solo podemos teorizar hacerca de lo que $T(G)$ es.
- Pero una vez que fijamos una teoría lingüística, intuitivamente $T(G)$ es tan important o más que $L(G)$.
- De alguna forma, podemos decir que $L(G)$ es solo el resultado de las características de $T(G)$.
- Resulta entonces raro que $T(G)$ sea solo algo 'derivado' en la gramática. Por que no tenemos gramáticas cuyo lenguaje sea $T(G)$.

Gramátias de Arboles

- En vez de construir árboles usando reglas de la gramática, representamos la regla directamente como un árbol
- Usaremos árboles elementales como unidades básicas que pueden utilizarse para reescribir no terminales en otros árboles elementales.
- La reescritura es equivalente a la expansion de un no-terminal en una gramática libre de contexto.

Arboles Finitos Ordenadios Etiquetados

- Un árbol finito ordenado etiquetado es una estructura $t = \langle N_t, \lhd_t, \prec_t, L_t \rangle$ donde

Arboles Finitos Ordenadios Etiquetados

- Un árbol finito ordenado etiquetado es una estructura $t = \langle N_t, \lhd_t, \prec_t, L_t \rangle$ donde
  - N_t es un conjunto finito no vacío de nodos

Arboles Finitos Ordenadios Etiquetados

- Un árbol finito ordenado etiquetado es una estructura $t = \langle N_t, \lhd_t, \prec_t, L_t \rangle$ donde
  - N_t es un conjunto finito no vacío de nodos
  - $\lhd_t$ y $\prec_t$ son relaciones binarias sobre N_t, respectivamente 'dominio' (dominance) y 'precedencia' (precedence)

Arboles Finitos Ordenadios Etiquetados

- Un árbol finito ordenado etiquetado es una estructura $t = \langle N_t, \lhd_t, \prec_t, L_t \rangle$ donde
  - N_t es un conjunto finito no vacío de nodos
  - $\lhd_t$ y $\prec_t$ son relaciones binarias sobre N_t, respectivamente 'dominio' (dominance) y 'precedencia' (precedence)
  - $\lhd^t$ define un árbol (i.e., un único nodo raíz y todo nodo excepto la raíz tiene un único predecesor) y $\prec_t$ define un orden total sobre los hijos de un nodo.

Arboles Finitos Ordenadios Etiquetados

- Un árbol finito ordenado etiquetado es una estructura $t = \langle N_t, \lhd_t, \prec_t, L_t \rangle$ donde
  - N_t es un conjunto finito no vacío de nodos
  - $\lhd_t$ y $\prec_t$ son relaciones binarias sobre N_t, respectivamente 'dominio' (dominance) y 'precedencia' (precedence)
  - $\lhd^t$ define un árbol (i.e., un único nodo raíz y todo nodo excepto la raíz tiene un único predecesor) y $\prec_t$ define un orden total sobre los hijos de un nodo.
  - L_t es la función de etiquetado, que asigna a cada nodo un conjunto de etiquetas.

Elementos de una TAG

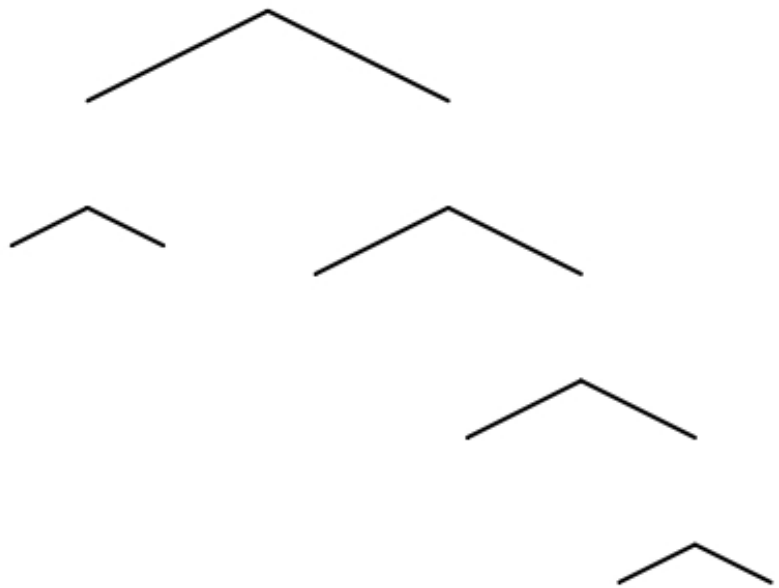
Una TAG es un conjunto finito de arboles finitos ordenados etiquetados con ciertas caracteristicas

- Tomemos dos conjuntos V_N (no terminales) y V_T (terminales), finitos no vacíos

Elementos de una TAG

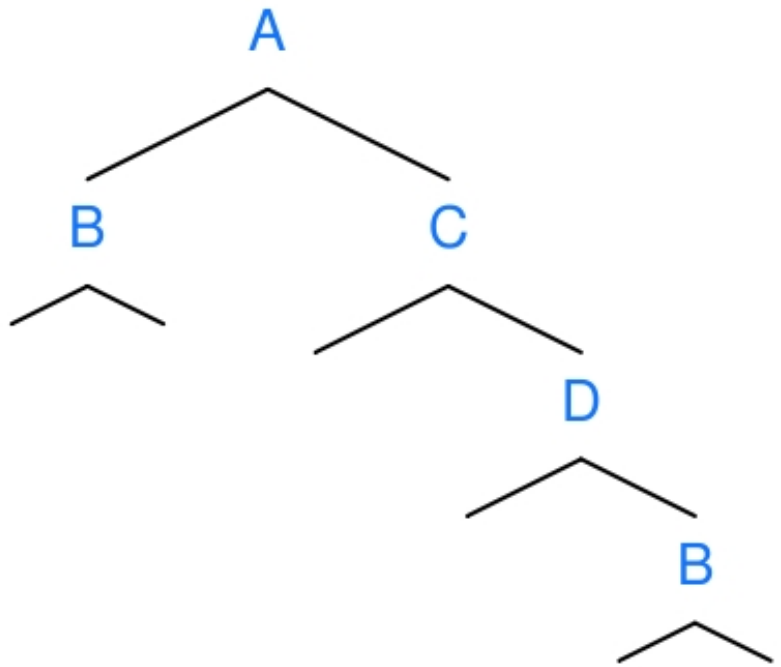
Una TAG es un conjunto finito de arboles finitos ordenados etiquetados con ciertas caracteristicas

- Tomemos dos conjuntos V_N (no terminales) y V_T (terminales), finitos no vacíos
- $t = \langle N_t, \lhd_t, \prec_t, L_t \rangle$



Elementos de una TAG

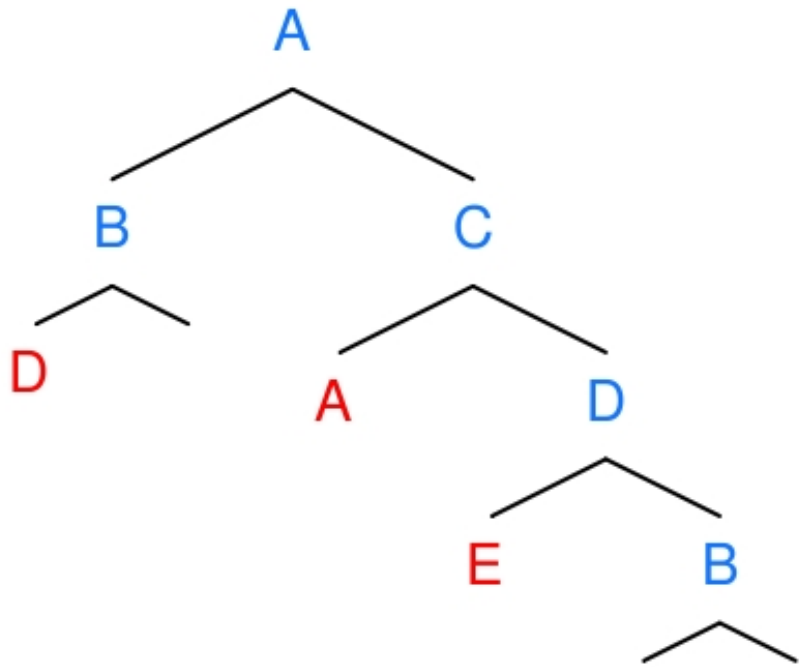
Una TAG es un conjunto finito de arboles finitos ordenados etiquetados con ciertas caracteristicas



- Tomemos dos conjuntos V_N (no terminales) y V_T (terminales), finitos no vacíos
- $t = \langle N_t, \lhd_t, \prec_t, L_t \rangle$
- L_t :
  $\text{NonLeaves}_t \rightarrow V_N$

Elementos de una TAG

Una TAG es un conjunto finito de arboles finitos ordenados etiquetados con ciertas caracteristicas



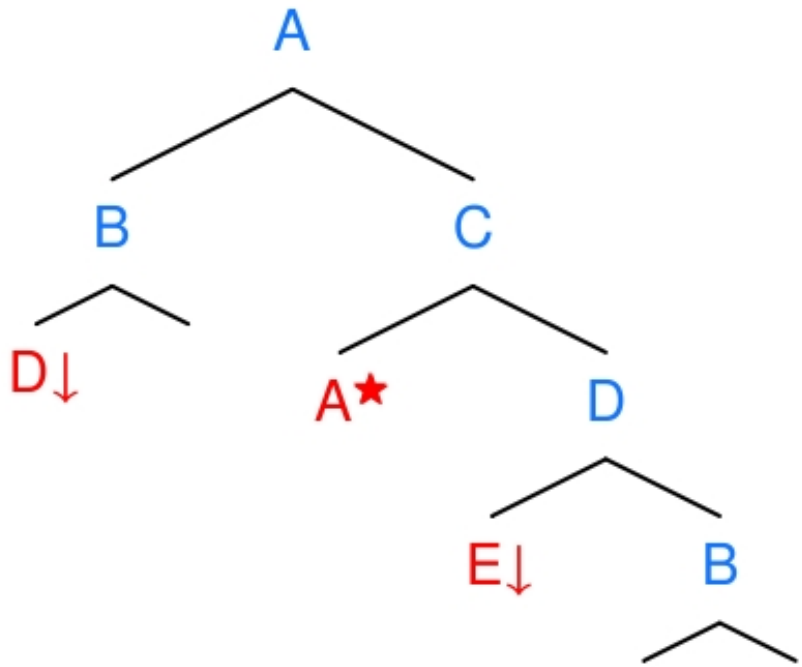
- Tomemos dos conjuntos V_N (no terminales) y V_T (terminales), finitos no vacíos
- $t = \langle N_t, \lhd_t, \prec_t, L_t \rangle$
- L_t :

  $\text{NonLeaves}_t \rightarrow V_N$

  $\text{Leaves}_t \rightarrow V_N$

Elementos de una TAG

Una TAG es un conjunto finito de arboles finitos ordenados etiquetados con ciertas caracteristicas



- Tomemos dos conjuntos V_N (no terminales) y V_T (terminales), finitos no vacíos
- $t = \langle N_t, \lhd_t, \prec_t, L_t \rangle$
- L_t :

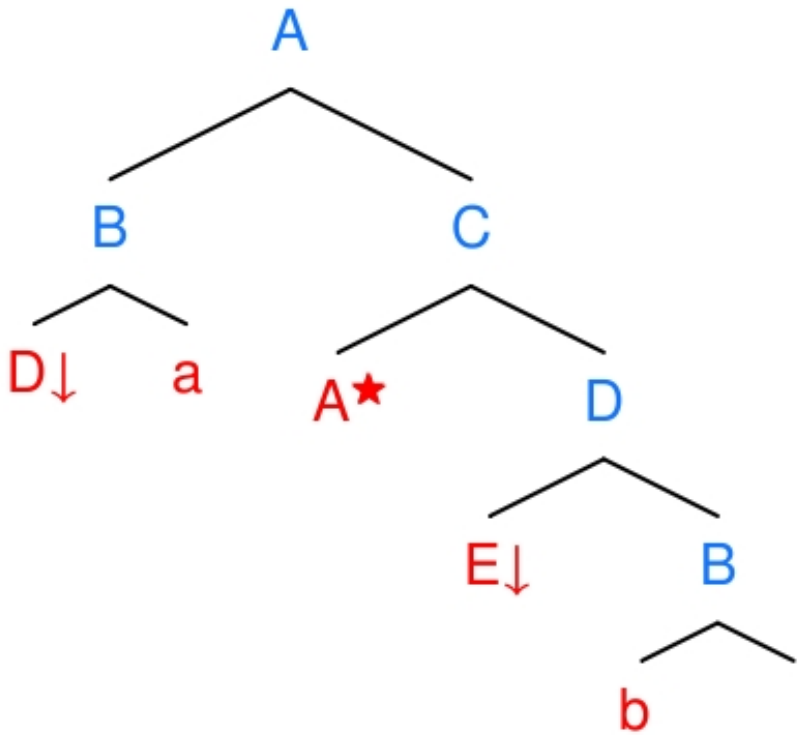
  $\text{NonLeaves}_t \rightarrow V_N$

  $\text{Leaves}_t \rightarrow V_N\{\downarrow, \star\}$

Elementos de una TAG

Una TAG es un conjunto finito de arboles finitos ordenados etiquetados con ciertas caracteristicas

- Tomemos dos conjuntos V_N (no terminales) y V_T (terminales), finitos no vacíos
- $t = \langle N_t, \lhd_t, \prec_t, L_t \rangle$
- L_t :
  $\text{NonLeaves}_t \rightarrow V_N$
  $\text{Leaves}_t \rightarrow V_N\{\downarrow, \star\} \cup V_T$



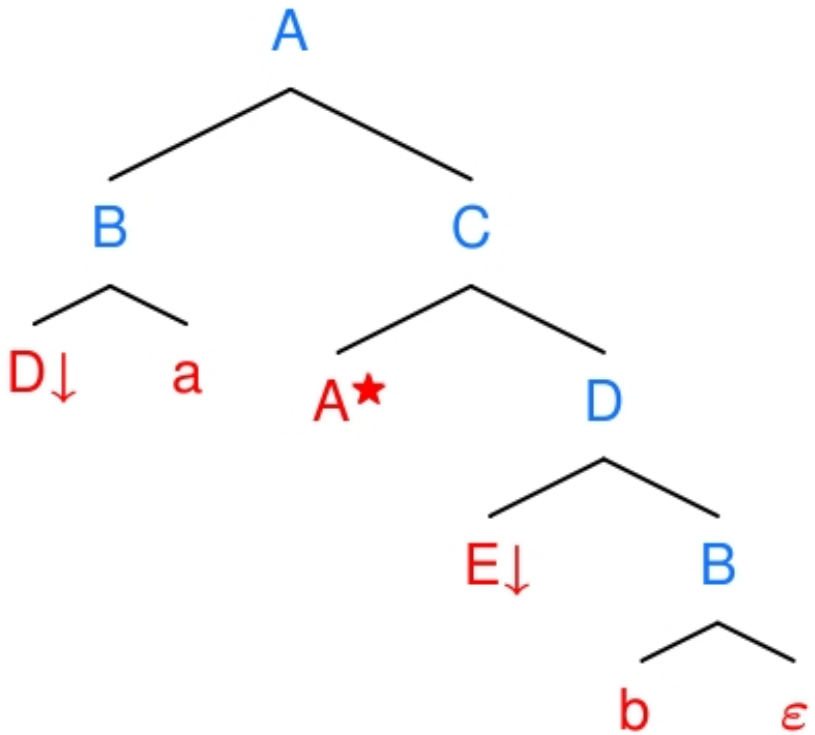
Elementos de una TAG

Una TAG es un conjunto finito de arboles finitos ordenados etiquetados con ciertas caracteristicas

- Tomemos dos conjuntos V_N (no terminales) y V_T (terminales), finitos no vacíos
- $t = \langle N_t, \lhd_t, \prec_t, L_t \rangle$
- L_t :

  $\text{NonLeaves}_t \rightarrow V_N$

  $\text{Leaves}_t \rightarrow V_N\{\downarrow, \star\} \cup V_T \cup \{\epsilon\}$

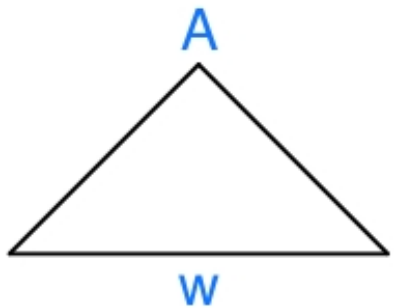


Derivaciones en TAG

- Podemos derivar nuevos arboles etiquetados mediante la aplicación de las operaciones de sustitución y adjunción.

Derivaciones en TAG

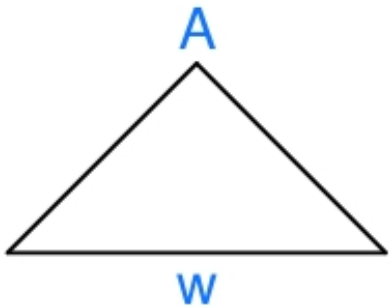
- Podemos derivar nuevos arboles etiquetados mediante la aplicación de las operaciones de sustitución y adjunción.
- Usando árboles iniciales



$A \in V_N$
$w \in (V_N\{\downarrow\} \cup V_T)^+$

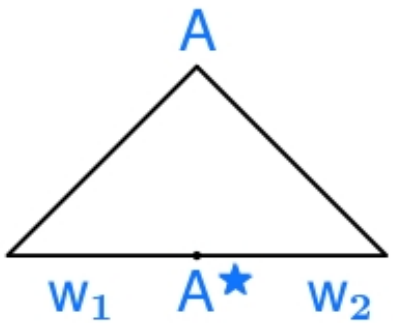
Derivaciones en TAG

- Podemos derivar nuevos arboles etiquetados mediante la aplicación de las operaciones de sustitución y adjunción.
- Usando árboles iniciales



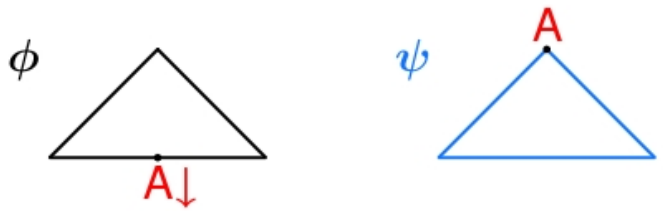
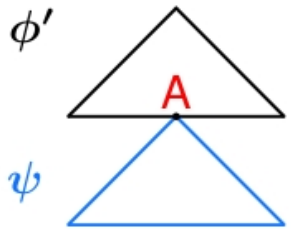
$A \in V_N$
$w \in (V_N\{\downarrow\} \cup V_T)^+$

- Y árboles auxiliares

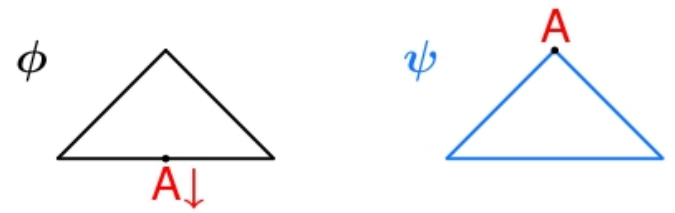
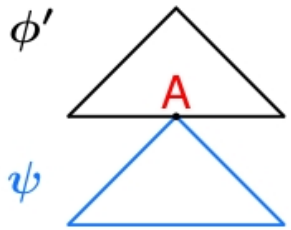


$A \in V_N$
$w_1, w_2 \in (V_N\{\downarrow\} \cup V_T)^+$

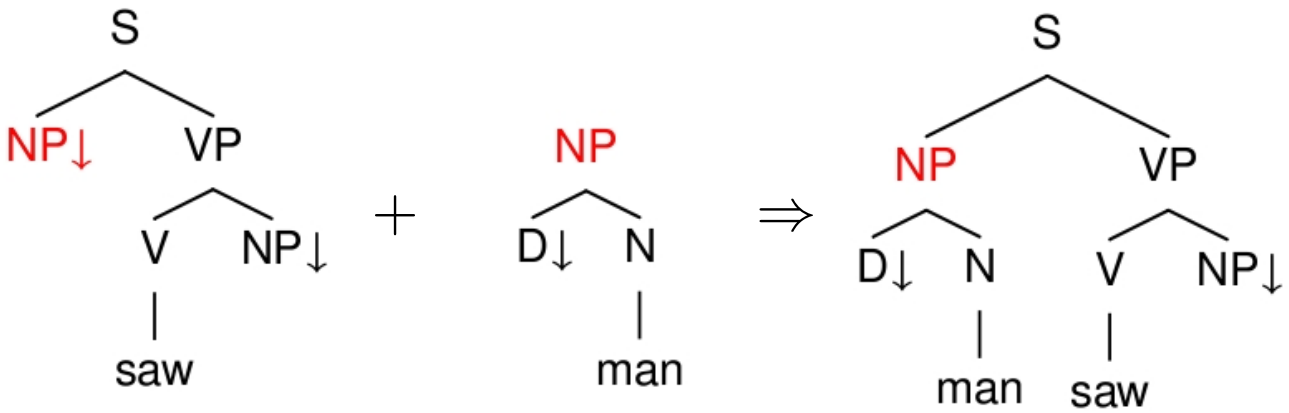
Substitución

From	We obtain
	
by substitución	

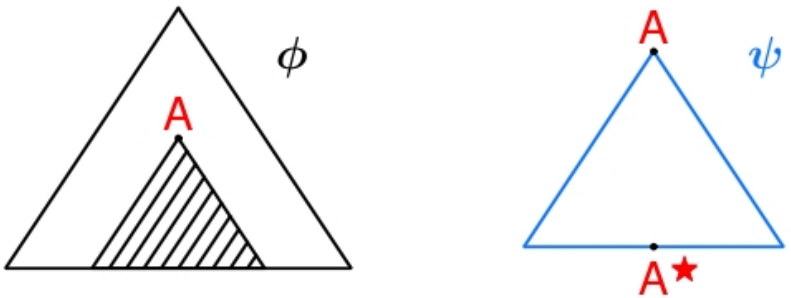
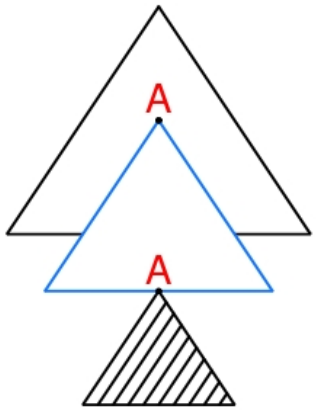
Substitución

From	We obtain
	
by substitución	

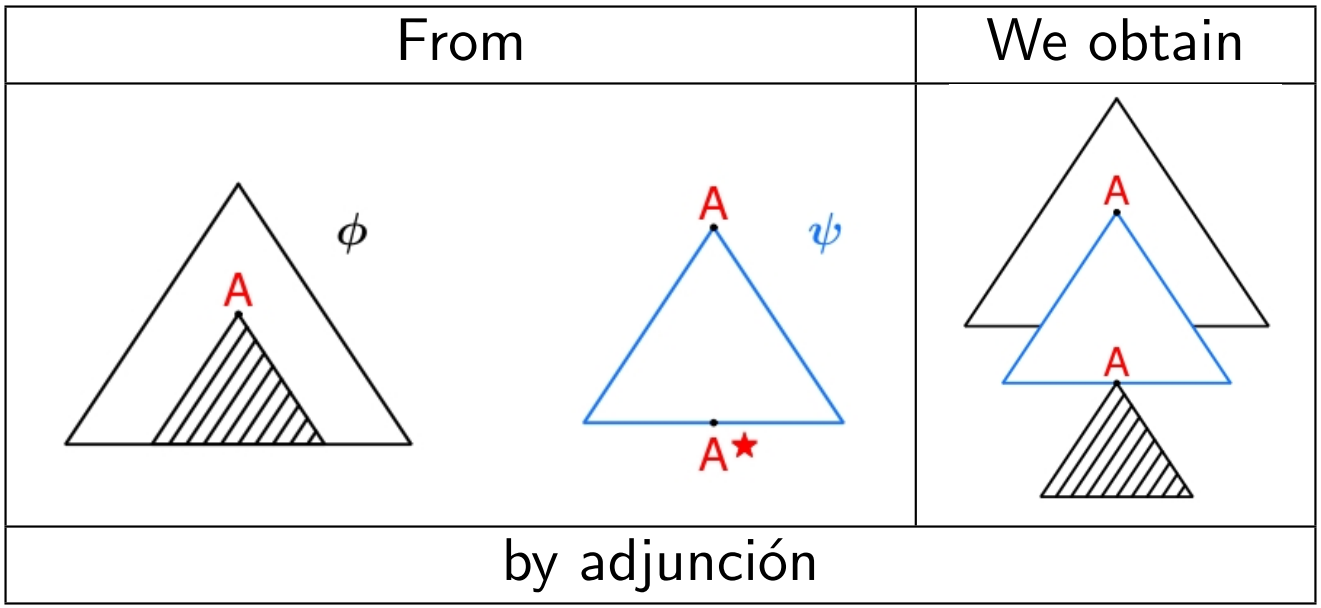
Ejemplo:



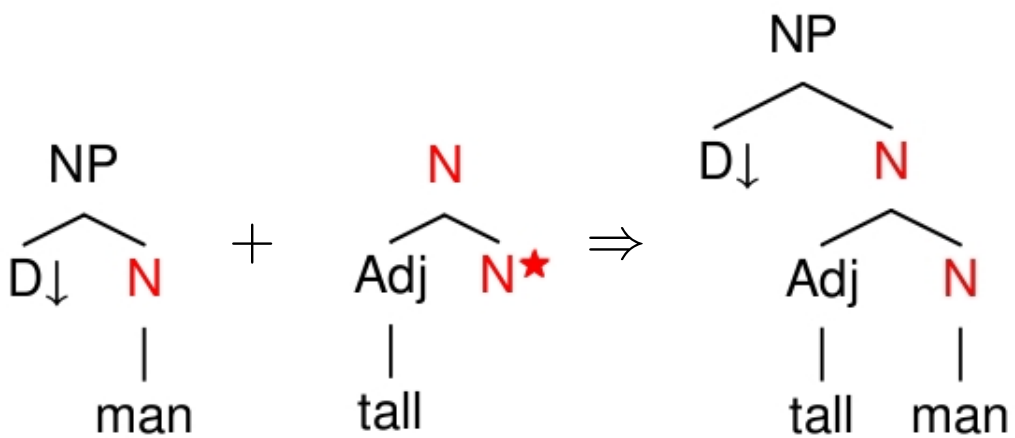
Adjunción

From	We obtain
	
by adjunción	

Adjunción



Ejemplo:



Tree Adjoining Grammar: Definición Formal

- Una Tree Adjoining Grammar es una tupla $G = (S, N, T, I, A)$ donde

Tree Adjoining Grammar: Definición Formal

- Una Tree Adjoining Grammar es una tupla $G = (S, N, T, I, A)$ donde
  - V_N es un conjunto de símbolos no terminales
  - V_T es un conjunto de terminales (disjunto de N)

Tree Adjoining Grammar: Definición Formal

- Una Tree Adjoining Grammar es una tupla $G = (S, N, T, I, A)$ donde
  - V_N es un conjunto de símbolos no terminales
  - V_T es un conjunto de terminales (disjunto de N)
  - I es un conjunto de árboles iniciales
  - A es un conjunto de árboles auxiliares

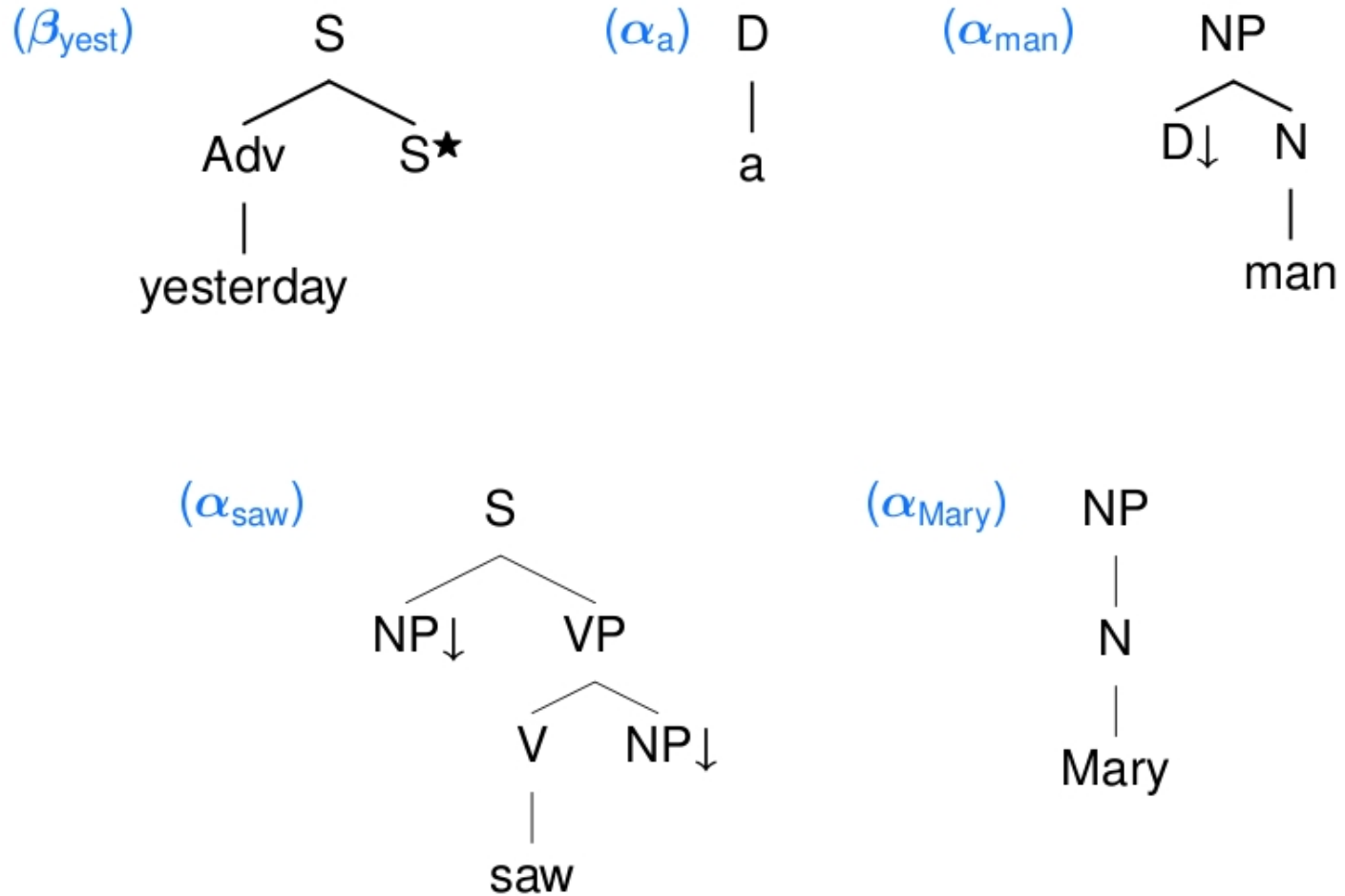
Tree Adjoining Grammar: Definición Formal

- Una Tree Adjoining Grammar es una tupla $G = (S, N, T, I, A)$ donde
  - V_N es un conjunto de símbolos no terminales
  - V_T es un conjunto de terminales (disjunto de N)
  - I es un conjunto de árboles iniciales
  - A es un conjunto de árboles auxiliares
  - $I \cup A$ es el conjunto de árboles elementales

Tree Adjoining Grammar: Definición Formal

- Una Tree Adjoining Grammar es una tupla $G = (S, N, T, I, A)$ donde
  - V_N es un conjunto de símbolos no terminales
  - V_T es un conjunto de terminales (disjunto de N)
  - I es un conjunto de árboles iniciales
  - A es un conjunto de árboles auxiliares
  - $I \cup A$ es el conjunto de árboles elementales
  - S (start) es un árbol inicial $S \in I$

Ejemplo


(β_{yest})
S
Adv
S★
yesterday
(α_a)
D
a
(α_{man})
NP
D↓
N
man
(α_{saw})
S
NP↓
VP
V
NP↓
saw
(α_{Mary})
NP
N
Mary

Lenguaje Generado por una TAG

- El lenguaje generado por una TAG:
  - es el conjunto de árboles que pueden obtenerse a partir del árbol inicial
  - utilizando las operaciones de sustitución y adjunción
  - tales que sus hojas son todos símbolos terminales.
- A partir del lenguaje de árboles de una TAG podemos obtener un lenguaje de strings considerando la 'frontera' (?)
- Extensiónes habituales:
  - Ciertos nodos no terminales pueden estar marcados como *na* indicando que una operación de adjunción no puede aplicarse en ese nodo
  - Ciertos nodos no terminales pueden estar marcados como permitiendo adjunciones sólo con cierto conjunto de árboles auxiliares.

Lo que Vemos Hoy

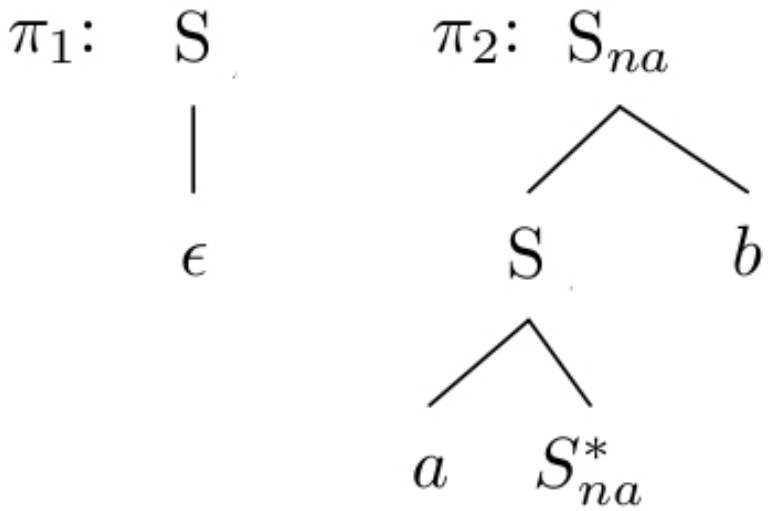
- Las Tareas Básicas de GLN
- Tree Adjoining Grammars (TAG)
- TAGs y Surface Realization

Capacidad Generativa Débil vs. Fuerte
Introduciendo TAG
Propiedades de TAG, Complejidad
Lexicalización de Gramáticas

Tree Adjoining Grammars

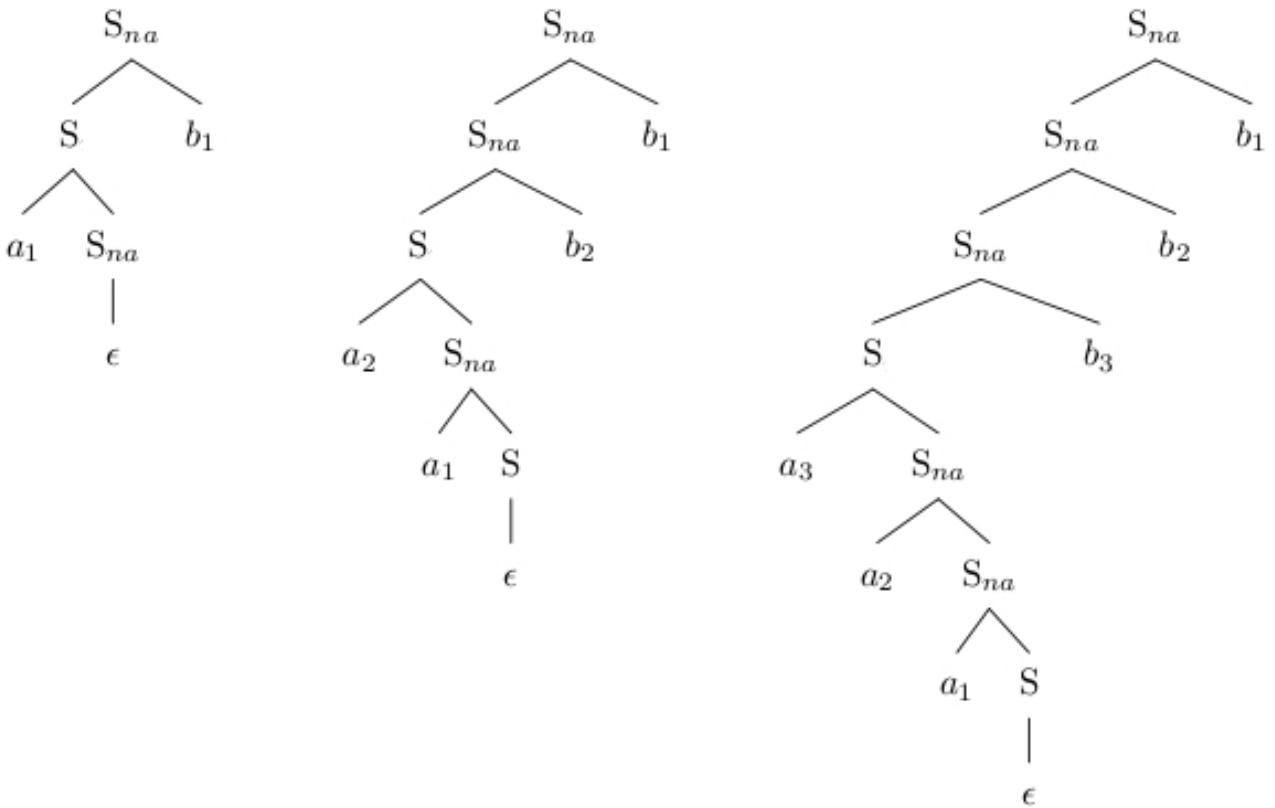
- Su poder expresivo está entre las gramáticas libres de contexto y las sensibles al contexto
- Pueden manejar todos los casos que mencionamos anteriormente de capacidad generativa débil y fuerte de lenguajes naturales.
- Permite modelar fácilmente casos de dependencias cruzadas y anidadas.

Dependencias Cruzadas en TAG: $a_2 a_1 b_2 b_1$

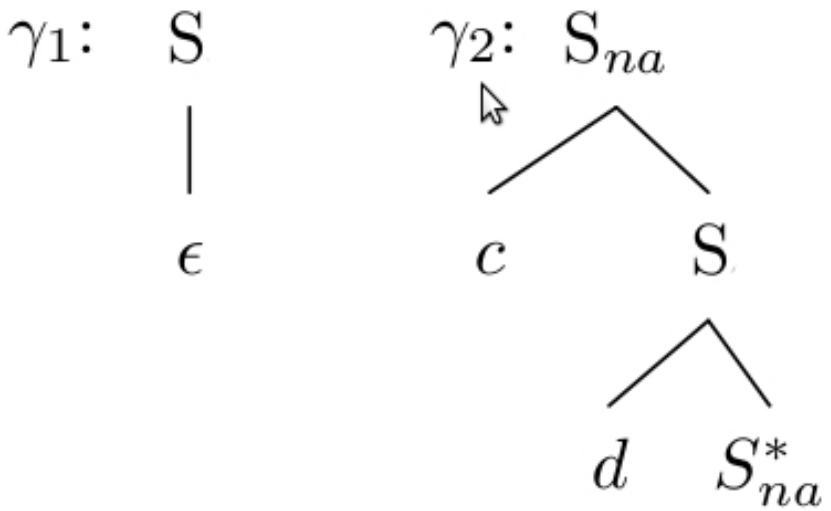


$$G : (T = \{a, b, \epsilon\}, N = \{S\}, I = \{\pi_1\}, A = \{\pi_2\}, S = \{\pi_1\})$$

Dependencias Cruzadas en TAG

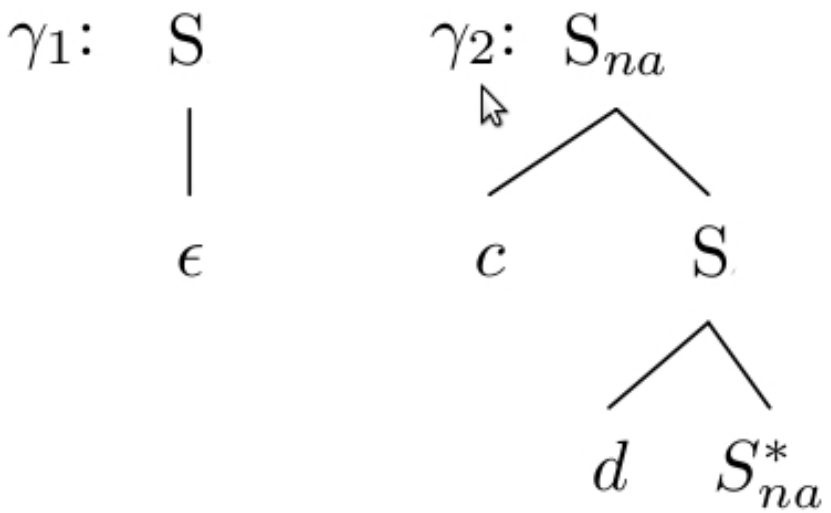


Dependencias Anidadas en TAG: $c_1 c_2 d_2 d_1$



$$G : (T = \{c, d, \epsilon\}, N = \{S\}, I = \{\gamma_1\}, A = \{\gamma_2\}, S = \{\gamma_1\})$$

Dependencias Anidadas en TAG: $c_1 c_2 d_2 d_1$



$$G : (T = \{c, d, \epsilon\}, N = \{S\}, I = \{\gamma_1\}, A = \{\gamma_2\}, S = \{\gamma_1\})$$

Ejercicio ver que podemos generar con una gramática que contenga π_1, π_2, γ_1, y γ_2?

Nos Ubicamos

- context-sensitive grammars:

$$0^i,\ i \text{ no es primo y } i > 0$$

Nos Ubicamos

- context-sensitive grammars:

$$0^i,\ i \text{ no es primo y } i > 0$$

- indexed grammars:

$$0^n 1^n 2^n \ldots m^n, \text{ para cualquier } m \text{ fijo y } n \geq 0$$

Nos Ubicamos

- context-sensitive grammars:

  0^i, i no es primo y $i > 0$

- indexed grammars:

  $0^n 1^n 2^n \ldots m^n$, para cualquier m fijo y $n \geq 0$

- tree-adjoining grammars (TAG):

  $0^n 1^n 2^n 3^n$, para $n \geq 0$

Nos Ubicamos

- context-sensitive grammars:
$$0^i,\ i \text{ no es primo y } i > 0$$
- indexed grammars:
$$0^n 1^n 2^n \ldots m^n, \text{ para cualquier } m \text{ fijo y } n \geq 0$$
- tree-adjoining grammars (TAG):
$$0^n 1^n 2^n 3^n, \text{ para } n \geq 0$$
- context-free grammars: $0^n 1^n$ para $n \geq 0$

Nos Ubicamos

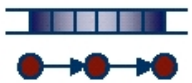
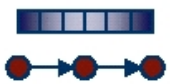
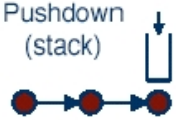
- context-sensitive grammars:
$$0^i,\ i \text{ no es primo y } i > 0$$
- indexed grammars:
$$0^n 1^n 2^n \ldots m^n, \text{ para cualquier } m \text{ fijo y } n \geq 0$$
- tree-adjoining grammars (TAG):
$$0^n 1^n 2^n 3^n, \text{ para } n \geq 0$$
- context-free grammars: $0^n 1^n$ para $n \geq 0$
- deterministic context-free grammars:

$$S \to \epsilon \mid Sc \mid SA \mid A,$$
$$A \to aSb \mid ab$$
el lenguaje de paréntesis balanceados

Nos Ubicamos

- context-sensitive grammars:
$$0^i,\ i \text{ no es primo y } i > 0$$
- indexed grammars:
$$0^n 1^n 2^n \ldots m^n, \text{ para cualquier } m \text{ fijo y } n \geq 0$$
- tree-adjoining grammars (TAG):
$$0^n 1^n 2^n 3^n, \text{ para } n \geq 0$$
- context-free grammars: $0^n 1^n$ para $n \geq 0$
- deterministic context-free grammars:
$$S \to \epsilon \mid Sc \mid SA \mid A,$$
$$A \to aSb \mid ab$$
el lenguaje de paréntesis balanceados
- regular grammars: $(0 \mid 1)^* 00 (0 \mid 1)^*$

Nos Ubicamos

Language	*Automaton*	*Grammar*	*Recognition*	*Dependency*
Recursively Enumerable Languages	Turing Machine 	Unrestricted Baa → A	Undecidable	Arbitrary
Context-Sensitive Languages	Linear-Bounded 	Context-Sensitive At → aA	NP-Complete 	Crossing 
Context-Free Languages	Pushdown (stack) 	Context-Free S → gSc	Polynomial 	Nested 
Regular Languages	Finite-State Machine 	Regular A → cA	Linear 	Strictly Local 

TAG esta entre CSL y CFG: puede ser reconocido en tiempo polinomial y puede representar dependencias cruzadas.

Complejidad

Problema: Dada una gramática G y un input x, determinar $x \in L(G)$

- unrestricted: indecidible

Complejidad

Problema: Dada una gramática G y un input x, determinar $x \in L(G)$

- unrestricted: indecidible
- context-sensitive: NSPACE[n] – linear non-deterministic space

Complejidad

Problema: Dada una gramática G y un input x, determinar $x \in L(G)$

- unrestricted: indecidible
- context-sensitive: NSPACE[n] – linear non-deterministic space
- indexed grammars: NP-Complete

Complejidad

Problema: Dada una gramática G y un input x, determinar $x \in L(G)$

- unrestricted: indecidible
- context-sensitive: NSPACE[n] – linear non-deterministic space
- indexed grammars: NP-Complete
- tree-adjoining grammars (TAG): $O(n^6)$

Complejidad

Problema: Dada una gramática G y un input x, determinar $x \in L(G)$

- unrestricted: indecidible
- context-sensitive: NSPACE[n] – linear non-deterministic space
- indexed grammars: NP-Complete
- tree-adjoining grammars (TAG): $O(n^6)$
- context-free: $O(n^3)$

Complejidad

Problema: Dada una gramática G y un input x, determinar $x \in L(G)$

- unrestricted: indecidible
- context-sensitive: NSPACE[n] – linear non-deterministic space
- indexed grammars: NP-Complete
- tree-adjoining grammars (TAG): $O(n^6)$
- context-free: $O(n^3)$
- deterministic context-free: $O(n)$

Complejidad

Problema: Dada una gramática G y un input x, determinar $x \in L(G)$

- unrestricted: indecidible
- context-sensitive: NSPACE[n] – linear non-deterministic space
- indexed grammars: NP-Complete
- tree-adjoining grammars (TAG): $O(n^6)$
- context-free: $O(n^3)$
- deterministic context-free: $O(n)$
- regular grammars: $O(n)$

Propiedades de TAG

- Pertenencia está en P (como dijimos es $O(n^6)$)

Propiedades de TAG

- Pertenencia está en P (como dijimos es $O(n^6)$)
- TALs (los lenguajes de string obtenidos mediante TAGs) estan cerrados por union, concatenación, clausura de Kleene, h, h^{-1}, intersección con RLs

Propiedades de TAG

- Pertenencia está en P (como dijimos es $O(n^6)$)
- TALs (los lenguajes de string obtenidos mediante TAGs) estan cerrados por union, concatenación, clausura de Kleene, h, h^{-1}, intersección con RLs
- TALs no están cerrados bajo intersección, intersección con CFLs ni complementación

Formalismos Gramaticales Restringidos

- Por que nos interesa definir el formalismo gramatical más restringido que nos permita modelar lenguaje natural?

Formalismos Gramaticales Restringidos

- Por que nos interesa definir el formalismo gramatical más restringido que nos permita modelar lenguaje natural?
- Y si no nos importa el costo computacional?
- Por ejemplo, un lingüísta podría usar el formalismo más general posible para analysar un lengauje, sin preocuparse de 'cuanto cuesta' computar ese modelo.

Formalismos Gramaticales Restringidos

- Por que nos interesa definir el formalismo gramatical más restringido que nos permita modelar lenguaje natural?
- Y si no nos importa el costo computacional?
- Por ejemplo, un lingüísta podría usar el formalismo más general posible para analysar un lengauje, sin preocuparse de 'cuanto cuesta' computar ese modelo.
  Si fuera necesario, podríamos despues ver si podemos traducir ('implementar') ese modelo a un formalismo computacionalmente eficiente.

Formalismos Gramaticales Restringidos

- En algunos casos, el formalismo gramatical nos puede ayudar proveyendo restricciones de que puede o no pasar.
- Si el formalismo esta cercano a cómo producimos el lenguaje, el mismo formalismo nos da pistas de como modelar cierto fenómeno.
- En otras palabras, un modelo erróneo debería ser mas díficil de escribir en un formalismo gramatical restringido, siempre y cuando ese formalismo este próximo a LA VERDAD.
- En otras, otras palabras, en un formalismo flexible es facil escribir tanto modelos correctos como incorrectos.

Lo que Vemos Hoy

- Las Tareas Básicas de GLN
- Tree Adjoining Grammars (TAG)
- TAGs y Surface Realization

Capacidad Generativa Débil vs. Fuerte
Introduciendo TAG
Propiedades de TAG, Complejidad
Lexicalización de Gramáticas

Gramáticas Lexicas

- Decimos que una gramatica G esta lexicalizada si cada regla de G contiene un terminal.

Gramáticas Lexicas

- Decimos que una gramatica G esta lexicalizada si cada regla de G contiene un terminal.
- Este tipo de gramáticas son especialmente útiles para parsing
  - al observar un terminal en el string a parsear podemos directamente restringir el conjunto de reglas de la gramática que pueden usarse para cubrirlo
- Pregunta: Dada una gramática arbitraria G, es posible encontrar una gramática G' lexicalizada tal que $L(G) = L(G')$?

Gramáticas Lexicas

- Decimos que una gramatica G esta lexicalizada si cada regla de G contiene un terminal.
- Este tipo de gramáticas son especialmente útiles para parsing
  - al observar un terminal en el string a parsear podemos directamente restringir el conjunto de reglas de la gramática que pueden usarse para cubrirlo
- Pregunta: Dada una gramática arbitraria G, es posible encontrar una gramática G' lexicalizada tal que $L(G) = L(G')$?
- Si G es una CFG la respuesta es afirmativa: usar la Forma Normal de Griebach

  Toda CFG puede ser reescrita de forma que sus reglas sean del tipo $A \to a\alpha$ donde a es un terminal.

Lexicalización de TAGs

- (Joshi and Schabes, 1997) muestran que TALs estas cerradas bajo lexicalización:

  toda TAL tiene una TAG lexicalizada que genera el mismo lenguaje.

Lexicalización de TAGs

- (Joshi and Schabes, 1997) muestran que TALs estas cerradas bajo lexicalización:

  toda TAL tiene una TAG lexicalizada que genera el mismo lenguaje.

- La idea de lexicalización es atractiva desde varias perspectivas
  - sintáxis
  - semántica (en Lingüística)
  - procesamiento y produccíon (en psicolingüística)

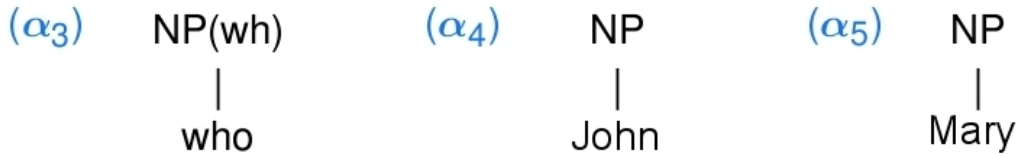
Lexicalización de TAGs

- (Joshi and Schabes, 1997) muestran que TALs estas cerradas bajo lexicalización:

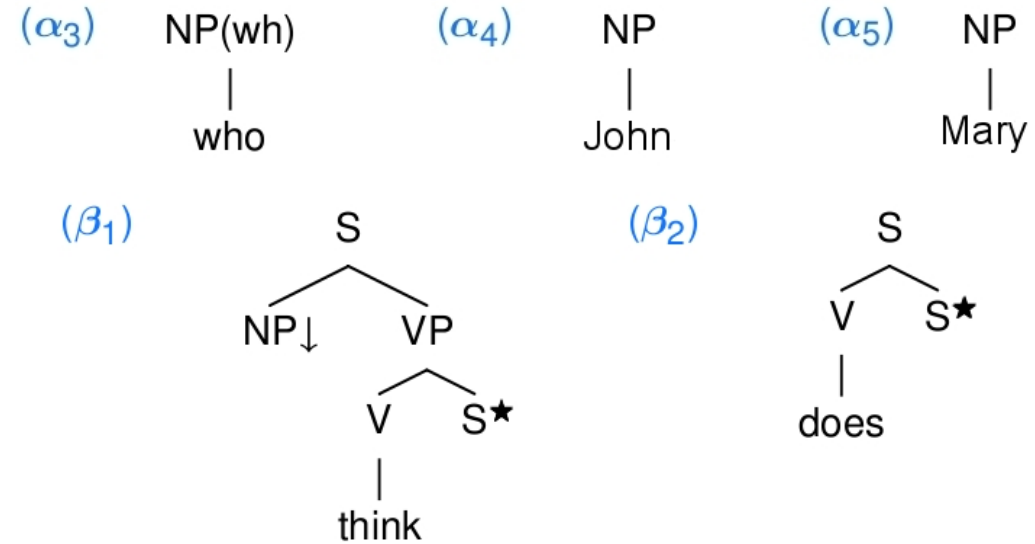
  toda TAL tiene una TAG lexicalizada que genera el mismo lenguaje.

- La idea de lexicalización es atractiva desde varias perspectivas
  - sintáxis
  - semántica (en Lingüística)
  - procesamiento y produccíon (en psicolingüística)
- Intuitivamente, lo que estamos diciendo es que cada palabra codifica un entorno local de restricciones sintácticas y semánticas.
- Muchos consideran esta intuición acertada y una TAG parece particularmente apropiada para capturarla.

Ejemplo: TAG Léxica



Ejemplo: TAG Léxica



Ejemplo: TAG Léxica