Clase #16 de 29
Niveles del Lenguaje
Parte I

Ago 12, Martes
Ago 20, Miércoles

Clase #17 de 29
Niveles del Lenguaje
Parte II

Ago 26, Martes
Ago 27, Miércoles

Los Niveles del Lenguaje

El Léxico, la Sintaxis, la Semántica & la Pragmática

Agenda para esta clase

- Niveles del Lenguaje
- Los Niveles en Castellano: “Vacunos voladores”
- Los Niveles en C: “Imprimir un valor”
- Pragmática en C: “Pop”

Los Niveles del Lenguaje

Niveles del Lenguaje

Nivel Pragmático

Nivel Semántico

Nivel Sintáctico

Nivel Léxico

Ejemplo en un Lenguaje Natural: El Castellano

Errores Léxicos

```
baka ~ buelan Loz
```

- No son palabras, no están en el diccionario
- Corrector ortográfico (*Spell Checker*) en Procesadores de Texto:
  - Dado el problema: Identificar si es palabra o no
    - ¿Cómo se resuelve?
      - Tabla de Símbolos
    - ¿Es "complejo"?
  - "Problema léxico":
    - ¿Es palabra?
    - ¿La secuencia de caracteres forman un lexema?

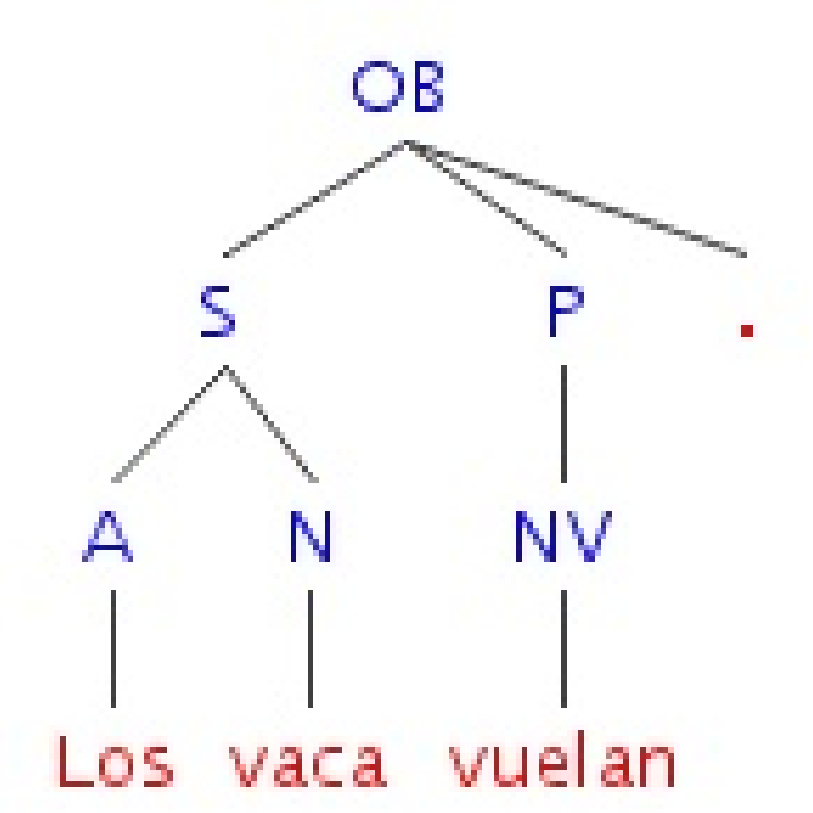
Errores Sintácticos

```
vaca . vuelan Los
```

- Ahora son palabras
  - ¿Por qué?
- … pero la *estructura* no es correcta, por lo tanto tampoco puede transmitir *significado* (semántica)
- Problema sintáctico: ¿Es palabra? ¿La secuencia de tokens tienen forman una escructura correcta?
- Reglas
  - Oración Bimembre:
    - *Sujeto Predicado* •
  - Sujeto:
    - *Artículo Núcleo*
  - Predicado:
    - *Núcleo Verbal*

Errores Semánticos

Los vaca vuelan.



- Reglas gramaticales
  - OB → S P •
  - S → A N
  - P → NV
- Ahora la estructura (sintaxis) es correcta
  - ¿Cuál es el árbol sintáctico ó árbol de derivación?
- … pero el género y número no coinciden, no expresa ningún significado
- Problema Semántico: ¿Es palabra? ¿La escructura sintácitca respeta las restricciones semánticas?.

Errores Pragmáticos

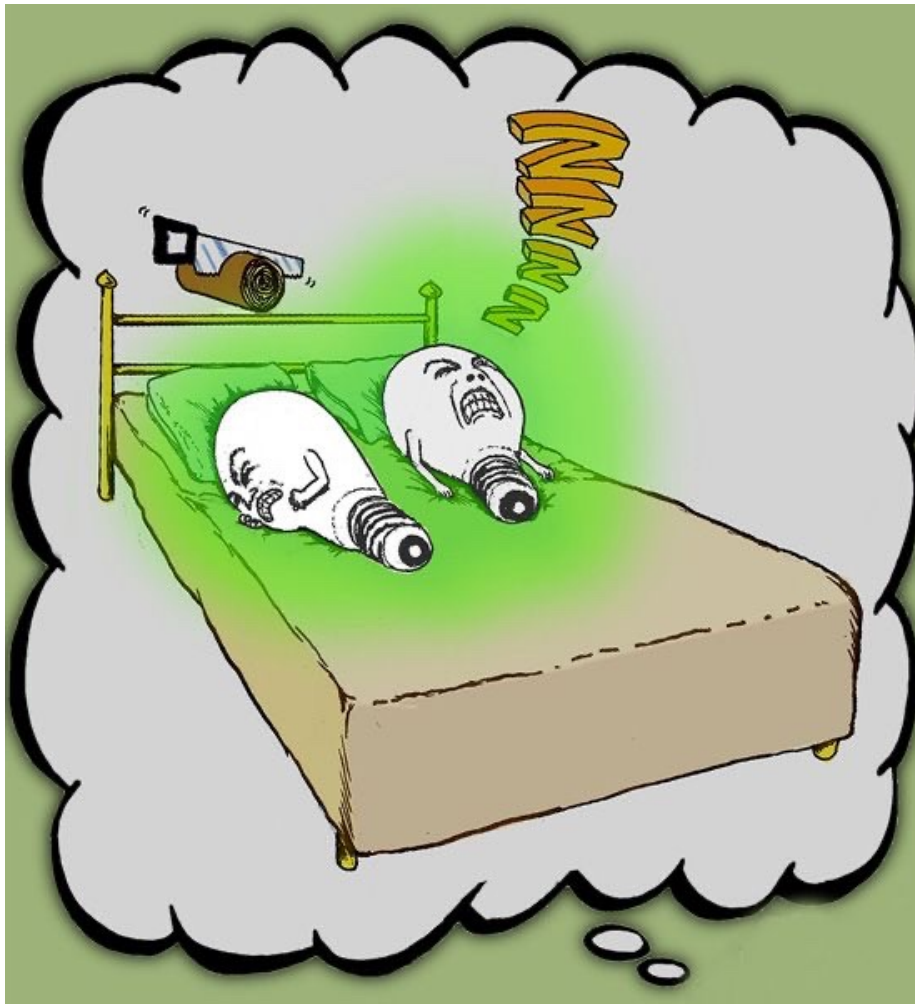
`Las vacas vuelan.`

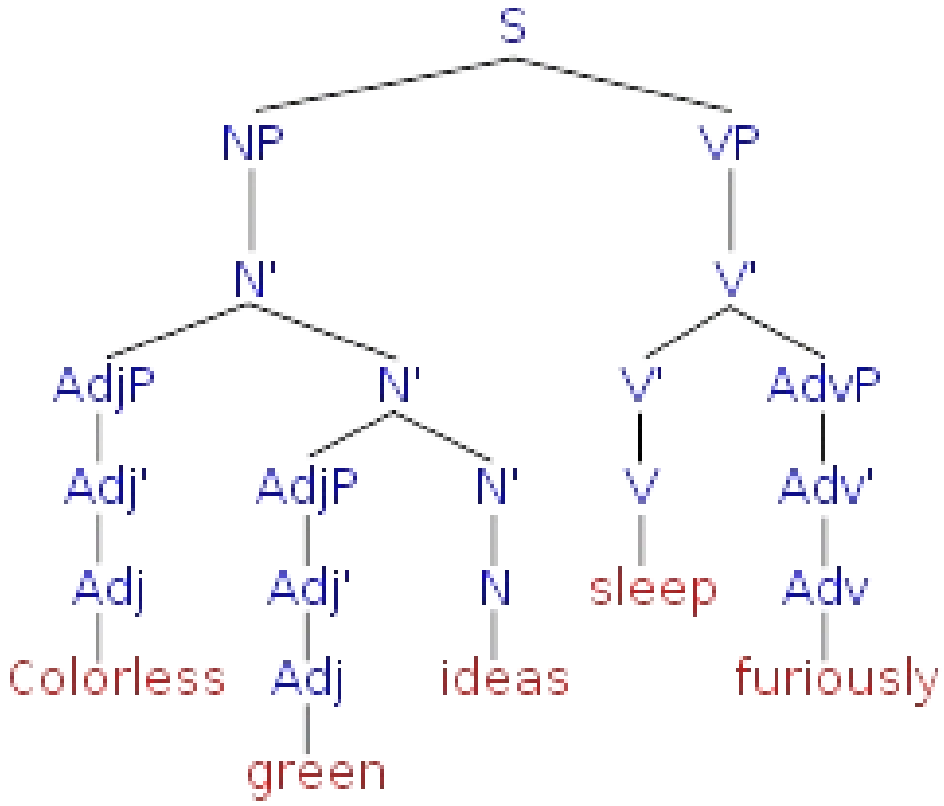
- Ahora expresa significado
  - ¿Por qué? ¿Según qué?
- … pero ¿es correcto?
- ¿Y si se lo usa en sentido figurativo para expresar imposibilidad?
- El error pragmático está relacionado con el uso.

Ambigüedades y complejidades de los lenguajes naturales

- Situación: una persona se sorprende y responde ante una pregunta de otra
  - **¿Cómo "¿cómo como?"?**
    ¡Como como como!
  - **¡Cómo como!**
- Análisis computacional
  - Buscador Web
    - Función del lexema
    - Tildes y acentos
  - IA.

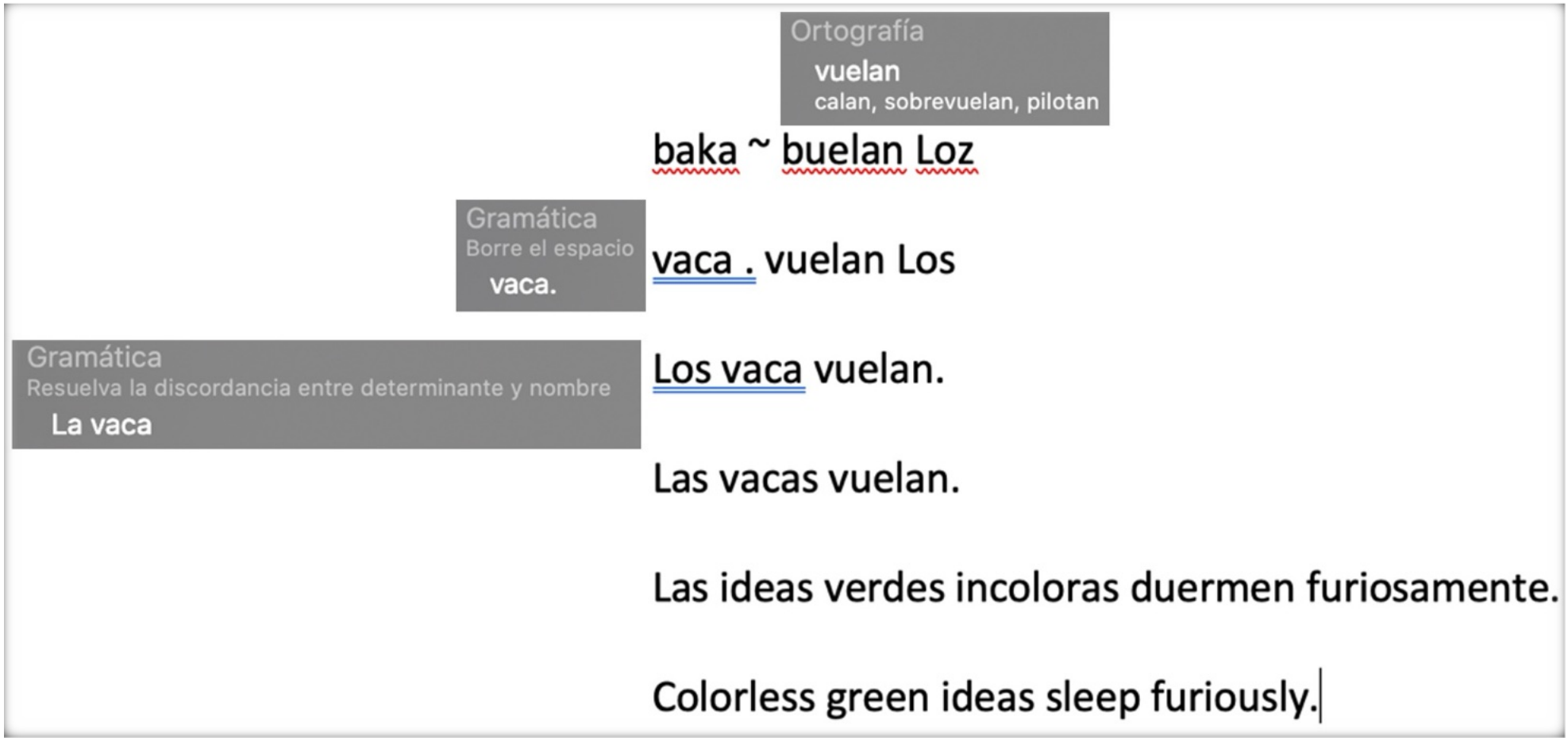
Noam Chomsky





Microsoft Word versus Chomsky

Microsoft Word for Mac 16.41 (2020)



Ejercicio

- Diseñe otro ejemplo del proceso en castellano; que inicie con errores léxicos, se resuelvan, queden errores sintácticos, se resuelvan, queden errores semánticos, y por último, se resuelvan. Describa cada error. Analice la pragmática de la frase resultante.

Ejemplo en un Lenguaje Formal: El Lenguaje de Programación C

Errores Léxicos:

Las subcadenas no siguen *patrones léxicos*, no son lexemas

```
Integer main(@)}
  Write "%d\n%c" 1..2);
  return 09
{
```

- ¿Cuál es el texto a analizar? ¿Es una sola cadena?
- ¿Cuáles son las subcadenas? ¿Cómo se las identifica?
- ¿Cuáles son los lexemas?
- ¿A qué categoría pertenecen?
  - Keywords (palabras claves)
    - Cardinalidad
    - ¿Cuál es la definición de este lenguaje?
  - Identificadores
    - ¿Cuál es la definición de este lenguaje?
    - Cardinalidad
    - Identificadores ó Palabras Reservadas
      - Cardinalidad
  - Literales
  - Cadenas Literales
  - Símbolos de Puntuación
- Errores léxicos detectados
  - Carácter inválido
  - Demasiados puntos decimales
  - Dígito octal inválido.

Errores Léxicos Corregidos:

Se pudo armar la secuencia de ***tokens***

```
Integer main(void)}
  Write "%d\n%c" a );
  return 0
{
```

tokens = ((*ID*,Integer), (*ID*, main), (*LPAR*), (*VOID*), (*RPAR*), (*RBRACE*), (*ID*, Write), (*LITERAL*,"%d\n%c"), (*ID*, a), (*RPAR*), (*SEMICOL*), (*RETURN*), (*LITERAL*, 0), (*LBRACE*))

- Subcadena
- Lexema
  - Patrones léxicos
- Token
  - Nombre y
  - Valor opcional
- Tokens simples
- Tokens “pares”
- Rresentación de tokens como números y referencias a tabla de símbols
- Secuencia de Tokens

Conceptos Lexicográficos y de Lenguajes Formales

- Carácter o símbolo
  - Alfabeto
  - $c \in \Sigma$
- Cadena (String)
  - Secuencia de Símbolos
  - $s \in \Sigma^*$
- Subcadena (String)
  - $p, q, s, t \in \Sigma^*$
  - $t = p \bullet s \bullet q$
- Lenguaje
  - $L \subset \Sigma^*$
- Lexema
  - Cadena que sigue un patrón determinado que forma un LF
  - $L \subset \Sigma^*$
- Token
  - Lexema que se le otorgó un significado
  - Tipos de tokens
    - Simples
    - Dobles, pares
  - Representación de tokens
    - Codificación del tipo
    - Codificación del valor

Errores Sintácticos:

Los tokens no forman estructuras según las reglas sintácticas

```
Integer main(void)}
  Write "%d\n%c" a );
  return 0
{
```

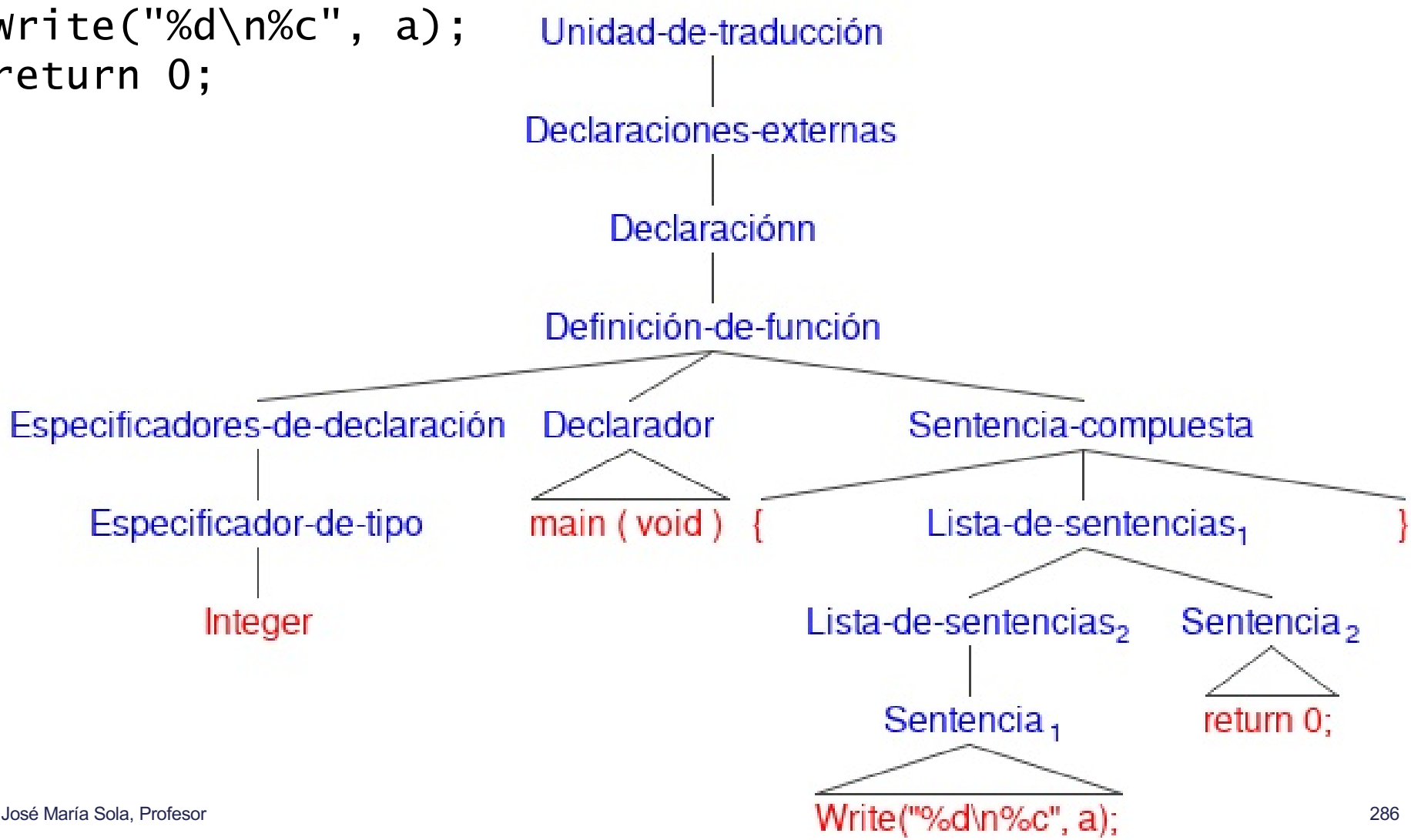
tokens = ((*ID*,Integer), (*ID*, main), (*LPAR*), (*VOID*), (*RPAR*), (*RBRACE*), (*ID*, Write), (*LITERAL*,"%d\n%c"), (*ID*, a), (*RPAR*), (*SEMICOL*), (*RETURN*), (*LITERAL*, 0), (*LBRACE*))

- Subcadena vs
- Lexema vs
- Token
  - Nombre y
  - Valor opcional
- Secuencia de Tokens
- Sentencia compuesta
  - no se cierra correctamente
- Expresión sufijo
  - La invocación carece de paréntesis y de la coma para separar argumentos
- Sentencia de salto
  - El punto y como no es opcional.
- Expresión
  - Correcta, es una Expresión Primaria

Errores Sintácticos Corregidos:

Con las reglas sintácticas se derivó el árbol sintáctico concreto

```
Integer main(void){
  Write("%d\n%c", a);
  return 0;
}
```



Errores Semánticos:

No se respetan las restricciones semánticas

```
Integer main(void){
  Write("%d\n%c", a);
  return 0;
}
```

- a.k.a, Errores Semánticos Estáticos
- Ni `Integer` ni `a` están declarados, esta UT no tiene significado para C
- ¿Qué ocurre con Write?.

Errores Semánticos Corregidos:

```
int main(void){
   int a;
   Write("%d\n%c", a);
   return 0;
}
```

Error en tiempo de Link (Enlace)

Hay referencias externas a la UT no resueltas

```
int main(void){
   int a;
   Write("%d\n%c", a);
   return 0;
}
```

- Caso Espacial `Write`:
- La función `Write` es una referencia externa (fuera de la UT) que no se puede resolver. Es decir, no se encuentra en la biblioteca Estándar
- Soluciones
  - Usar una biblioteca en particular
  - Usar una función estándar.

Error en tiempo de Link Corregido

```
int main(void){
   int a;
   printf("%d\n%c", a);
   return 0;
}
```

- ¿Funciona?
- Comportamiento indefinido:
  - printf *pop*ea un argumento extra a la cantidad de argumentos enviados.
  - ¿Ayuda el protipo acá?
- Veamos los diferentes tipos de comportamiento, que arman una jerarquía en base a su determinismo.

Jerarquía de Comportamientos

Semántica en Ejecución

Semántica y Comportamiento

- **Comportamiento (behavior) Definido**
  - Apariencia o acción externa explicitado por la especificación del lenguaje
    - e.g. *break;*
- **Comportamiento específico a locación (locale-specific behavior)**
  - Comportamiento que depende de nacionalidad, cultura y lenguaje.
    - e.g. *islower*
- **Comportamiento definido por la implementación (implementation-defined behavior)**
  - Es un comportamiento no especificado, pero cada implementación documenta su elección.
    - e.g. *cantidad de caracteres significativos en un identificador; corrimiento a derecha de bits en enteros signados.*
- **Comportamiento No Especificado (unspecified behavior)**
  - Dos o más posibilidades, pero sin ninguna restricción.
    - e.g. *orden de evaluación de argumentos.*
- **Comportamiento Indefinido (undefined behavior)**
  - Sin requisitos del estándar sobre la implementación: "Here be dragons" "HIC SUNT DRACONES"
  - Rango de comportamiento válido: **ignorar** con resultados impredecibles, **diagnosticar**, comportarse según alguna **documentación** de la implementación, **terminar** traducción o ejecución.
    - e.g. *Subindicación fuera de rango del arreglo.*

Comportamiento Indefinido I:

Casos sin requisitos de comportamiento

(a.k.a. Error Semántico Dinámico)

```
{ char s[]={'a', 'b', 'c', 'd'};
  puts(s);
  s[4]='\0';
  int GetIndex(void);
  s[GetIndex()]='\0';
}

{ int n=7, f();
  printf( "%d", n/f() );
}
```

- Hay muchos más, son los **comportamientos no definidos** por el lenguaje.

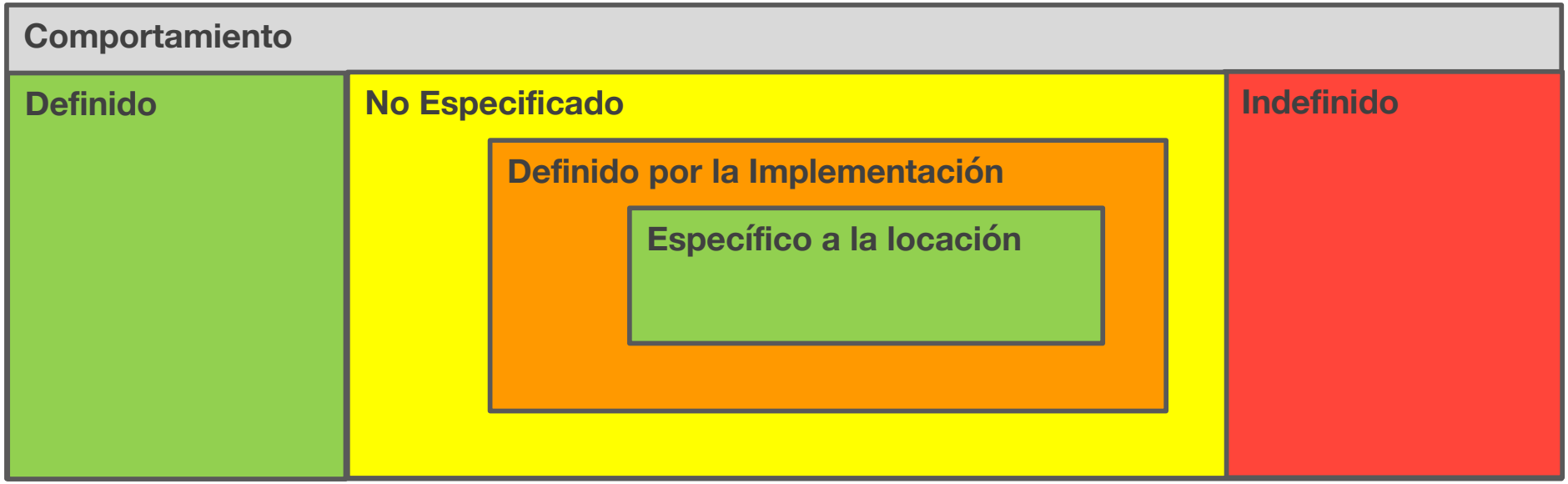
Comportamiento Indefinido II:

Precondiciones de las Funciones

```
int main(){
   int a;
   printf("%d\n%c", a);
   return 0;
}
```

- Se invoca a `printf` con dos argumentos
- Al comenzar, `printf` toma el primero
- Al haber dos valores para formatear , `printf` intenta obtener dos valores, pero solo hay uno disponible
- **¿Ayuda el prototipo en este caso?**
  - ¿Qué errores evita usar prototipos?.

Jerarquía de Comportamientos



- Actividades propuestas
  - Escriba tres ejemplos en C de cada tipo de comportamiento, justifique.
  - Investigue y explique la razón de existencia de cada tipo de comportamiento.
  - Explique y ejemplifique las implicanciads de basar el comportamiento de nuestro programa en cada uno de los tipos de comportamiento.
  - Investigue y comente qué otros LP, diferentes a C y C++, tienen estos conceptos y como los aplica.
  - Investigue y explique el nuevo *Comportamiento Erróneo* de C++
  - Explique ventajas y riesgos de cada tipo de comportamiento, analice la *portabildad*.

Comportamiento Indefinido *Evitado*

```
#include <stdio.h>
int main(void){
  int a;
  printf("%d\n", a);
  return 0;
}
```

Error Pragmático

El comportamiento no es el esperado

```
#include <stdio.h>
int main(void){
  int a;
  printf("%d\n", a);
  return 0;
}
```

```
#include <stdio.h>
int main(void){
  int a=0x2A;
  printf("%d\n", a);
}
```

```
#include <stdio.h>
int main(void){
  puts("42");
}
```

- Se usa la variable `a` sin estar inicializada
- ¿Pero si la intención del programador es, justamente, ver el valor basura de la variable `a`?
- ¿Es necesario el `return`?

Ejercicio

- Diseñe otro ejemplo del proceso en C; que inicie con errores léxicos, se resuelvan, queden errores sintácticos, se resuelvan, queden errores semánticos, y por último, se resuelvan. Describa cada error. Analice la pragmática del programa resultante.
- Diseñe otro ejemplo del proceso en otro LP que no sea C ni C++.

Estilo y Expresiones Idiomáticas

Caso de Estudio de Pragmática: Pop()

Pop y Expresiones Idiomáticas

```
static int theElements[MAX];
static size_t theLevel;
```

```
int Pop(){
  return theElements[--theLevel];
}
```

```
int Pop(){
  int e;
  e = theElements[ theLevel - 1 ];
  theElements[ theLevel - 1 ] = 0;
  theLevel = theLevel - 1;
  return e;
}
```

Análisis Comparativo

```
int Pop(){
  int e;
  e = theElements[ theLevel-1 ];
  theElements[ theLevel - 1 ] = 0;
  theLevel = theLevel - 1;
  return e;
}
```

- Objetivo
  - Tiempo
    - Traducción
    - Ejecución
      - Accesos
      - Evaluaciones
      - Asignaciones
      - Restas
  - Espacio
    - Código fuente en disco
      - Línea de código
    - Código objeto en memoria
      - Optimización del compilador
    - Objetos en memoria
- Subjetivo
  - Estilo correcto
  - Estilo claro
  - Mantenimiento
  - Depuración
  - Proabilidad de ocurrencia de bug

```
int Pop(){
  return theElements[--theLevel];
}
```

- Otra implementación contigua

```
static int theElements[MAX];
static int *top = theElements;
```

```
int Pop(){
  return *--top;
}
```

- Puntero vs. Índice
  - Equivalencia de expresiones
- Aplicación de:
  - Predecremento
  - Aritmética de punteros
- Analizar
  - Incialización de p
  - Accesos
  - Evaluaciones
  - Arreglo

Selección del Nombre y Alcance de los Identificadores

```
// StackModule.h
int Pop();
```

```
// StackModule.c - Subindicación
static int theElements[MAX];
static size_t theLevel;

int Pop(){
  return theElements[--theLevel];
}
```

```
// StackModule.c - Subindicación
static int a[MAX];
static size_t n;

int Pop(){
  return a[--n];
}
```

```
// StackModule.c - Puntero
static int theElements[MAX];
static int *top = theElements;

int Pop(){
  return *--top;
}
```

```
// StackModule.c - Puntero
static int a[MAX];
static int *p = a;

int Pop(){
  return *--p;
}
```

- ¿Cuál es el alcance de cada identificador?
- ¿Hay analogía con la POO?

- Selección del Identificador de la Entidad
  - Entidad Privada
    - Alance reducido
    - Longitud de nombre reducido
    - Representatividad baja
  - Entidad Pública
    - Alance extendido
    - Longitud de nombre extendido
    - Representividad alta.

Trabajo Opcional: Diseño de Stack

- Diseñe e implemente por lo menos tres variantes de Stack según las siguientes variables
  - *Contigua* versus *Enlazada*
  - *Con Precondiciones* versus Sin Precondiciones
  - *Elementos de tipo Genérico* versus *Elementos de tipos Concretos*
  - *Módulo* versus *Tipo de Dato*
  - *Semántica de Referencia* versus *Semántica de Valor*
  - *Funcionales* (i.e., funciones puras) versus *Con efecto de lado*
  - *Con Capacidad Máxima* versus *Sin Capacidad Máxima*
- Operaciones Primitivas
  - *Push*
  - *Pop*
- Operaciones Derivadas
  - *Top* ó *Peek*
  - *IsEmpty*
  - *Clear* ó *Empty*
  - *GetSize* ó *GetLength*
- Restricciones
  - Las operaciones deben ser O(1), salvo *Clear* o *Empty* que en algunas implementaciones puede llegar a O(n)
- Entrega
  - Especificación del Tipo o Módulo
  - Interfaz
  - Pruebas
  - Implementación
  - Característica sobresaliente de la implementación y fundamento de la selección, por ejemplo:
    - "Esta implementación de Stack se diseñó para ser usada en el espacio exterior por personas de sagitario" ó
    - "Esta implementación de Stack se diseñó para ser un contenedor eficiente en tiempo de caracteres que pueden aparecer en Scrabble"

Términos de la clase #15-16 Parte II

Definir cada término con la bibliografía

- Niveles del Lenguaje
- Nivel Léxico
- Nivel Sintaxis
- Nivel Semántico
- Nivel Pragmático
- Error Léxico
- Problema léxico
- Error Sintáctico
- Reglas sintácticas o Gramática
- Problema sintáctico
- Problema semántico
- Árbol sintáctico ó Árbol de Derivación
- Ambigüedades y complejidades de los lenguajes naturales
- Noam Chomsky
- Lexema
- Patrón Léxico
- Categoría Léxica
- Keyword
- Identificadores
- Literales
- Cadenas Literales
- Símbolos de Puntuación
- Token
- Sentencia Compuesta
- Expresión Sufijo
- Sentencia de Salto
- Expresión Primaria
- Unidad de traducción
- Declaración Externa
- Declaración
- Definición de Función
- Especificadores de Declaración
- Declarador
- Árbol sintáctico con Sintaxis Concreta
- Error Semántico estático
- Error en tiempo de Link (Enlace)
- Referencia Externa
- Prototipo
- Semántica y Comportamiento
- Comportamiento (behavior)
- Comportamiento No Especificado (unspecified behavior)
- Comportamiento definido por la implementación (implementation-defined behavior)
- Comportamiento específico a locación (locale-specific behavior)
- Comportamiento Indefinido (undefined behavior)
- Error Semántico dinámico
- Comportamiento Indefinido
- Error Pragmático
- Selección del alcance de los identificadores.

Tareas para la próxima clase

1. (*Opcional*) Generar un informe con los mensajes de diagnóstico de su compilador para cada versión del programa que imprime un valor, pasar por todos los niveles y sacar sus propias conclusiones.

¿Consultas?

Fin de la clase