

Clase #15 de 27

Los Niveles del Lenguaje

El Léxico, la Sintaxis, la Semántica & la Pragmática

Esp. Ing. José María Sola, profesor

2021 Sept 6, 13

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 ~ bue1an 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?

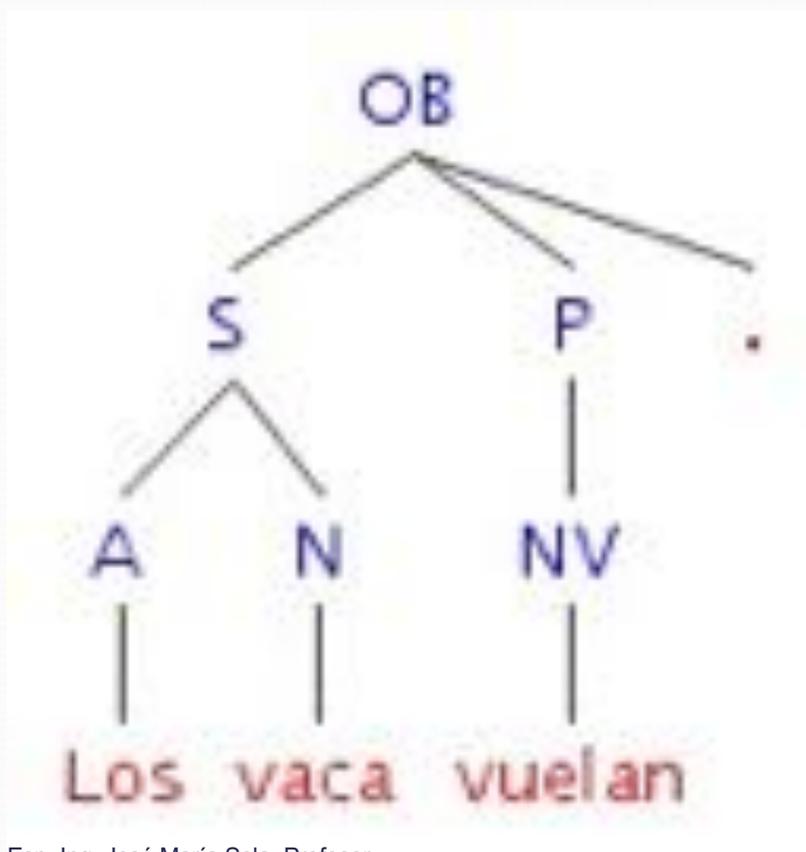
Errores Sintácticos

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 forma una estructura correcta?
- Reglas
 - Oración Bimembre:
 - *Sujeto Predicado* •
 - Sujeto:
 - *Artículo Núcleo*
 - Predicado:
 - *Núcleo Verbal*

Errores Semánticos

Los vaca vuelan.



- Reglas gramaticales
 - $OB \rightarrow S P \cdot$
 - $S \rightarrow A N$
 - $P \rightarrow NV$
- Ahora la estructura (syntax) 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 estructura sintáctica respeta las restricciones semánticas?.

Errores Pragmáticos

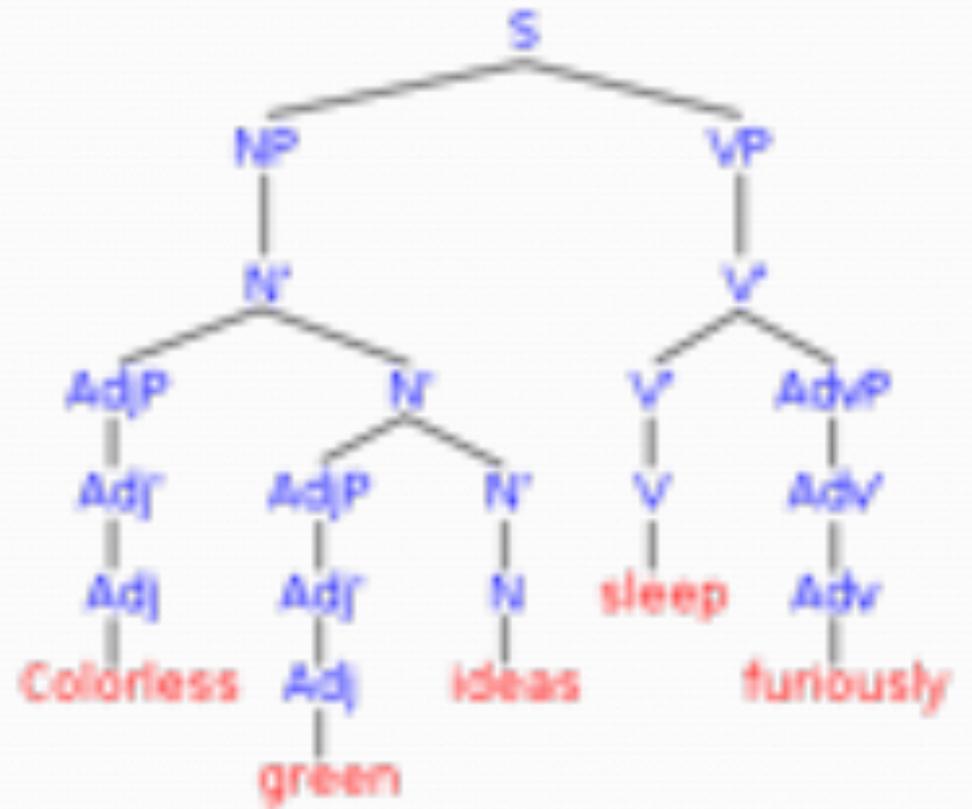
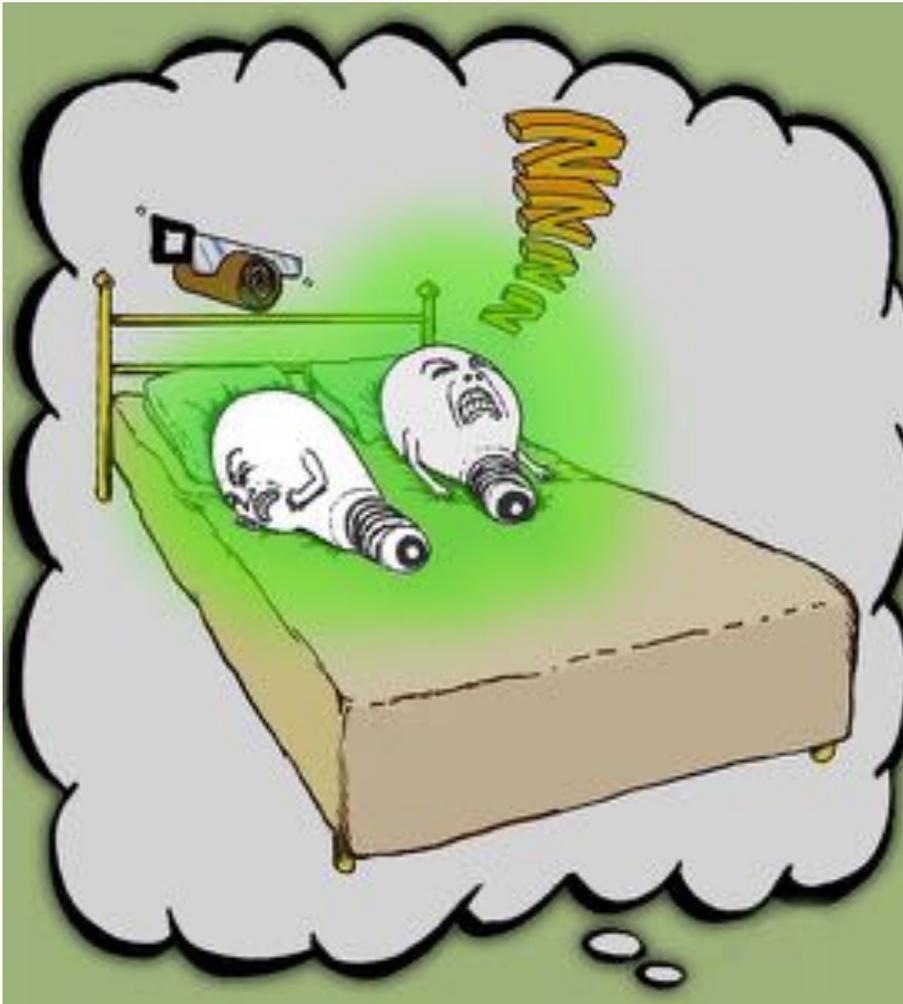
Las vacas vuelan.

- 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.

Noam Chomsky



Microsoft Word versus Chomsky

Microsoft Word for Mac 16.41 (2020)

The screenshot shows a Microsoft Word document with several grammar and spelling suggestions. The text in the document is as follows:

Los vaca vuelan.

Las vacas vuelan.

Las ideas verdes incoloras duermen furiosamente.

Colorless green ideas sleep furiously.

The suggestions are shown in grey boxes:

- Ortografía**
vuelan
calan, sobrevuelan, pilotan
- Gramática**
Borre el espacio
vaca.
- Gramática**
Resuelva la discordancia entre determinante y nombre
La vaca

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 se ajustan a los 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 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), (*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

Estructura Sintáctica: Árbol de Derivación con Sintaxis Concreta



Errores Semánticos:

No se respetan las restricciones semánticas
(a.k.a, Errores Semánticos Estáticos)

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

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

Error en tiempo de Link (Enlace)

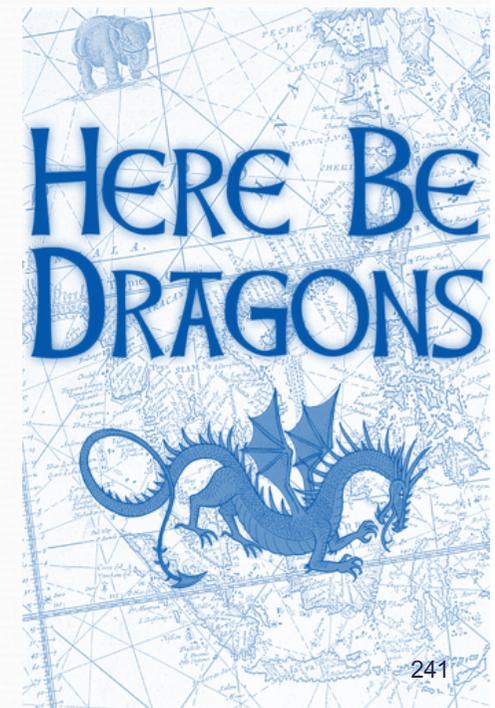
Hay referencias externas a la UT no resueltas

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

- Caso Especial `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.

Semántica y Comportamiento

- **Comportamiento (behavior)**
 - Apariencia o acción externa
 - e.g. *break*
- **Comportamiento No Especificado (unspecified behavior)**
 - Dos o más posibilidades, pero sin ninguna restricción.
 - e.g. *orden de evaluación de argumentos.*
- **Comportamiento definido por la implementación (implementation-defined behavior)**
 - Es un comportamiento no especificado, pero cada implementación documenta su elección.
 - e.g. *corrimiento a derecha de bits en enteros signados.*
- **Comportamiento específico a locación (locale-specific behavior)**
 - Comportamiento que depende de nacionalidad, cultura y lenguaje.
 - e.g. *islower*
- **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.*



Comporamiento Indefinido I:

Precondicciones de las Funciones

```
int main(void){  
    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?.

Comportamiento Indefinido II:

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, f(void);
  n/f();
}
```

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

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 de la frase 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(void){  
    return theElements[--theLevel];  
}
```

```
theElements[ theLevel - 1 ] = 0;
```

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

Análisis Comparativo

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

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

- Otra implementación contigua

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

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

- 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

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

Selección del Nombre y Alcance de los Identificadores

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

```
// StackModule.c - Subindicación  
static int theElements[MAX];  
static size_t theLevel;  
  
int Pop(void){  
    return theElements[--theLevel];  
}
```

```
// StackModule.c - Subindicación  
static int a[MAX];  
static size_t n;  
  
int Pop(void){  
    return a[--n];  
}
```

```
// StackModule.c - Puntero  
static int theElements[MAX];  
static int *top = theElements;  
  
int Pop(void){  
    return *--top;  
}
```

```
// StackModule.c - Puntero  
static int a[MAX];  
static int *p = a;  
  
int Pop(void){  
    return *--p;  
}
```

- ¿Cuál es el alcance de cada identificador?
- ¿Hay analogía con la POO?
- Selección del Identificador de la Entidad
 - Entidad Privada
 - Alcance reducido
 - Longitud de nombre reducido
 - Representatividad baja
 - Entidad Pública
 - Alcance extendido
 - Longitud de nombre extendido
 - Representatividad alta.

Trabajo: 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