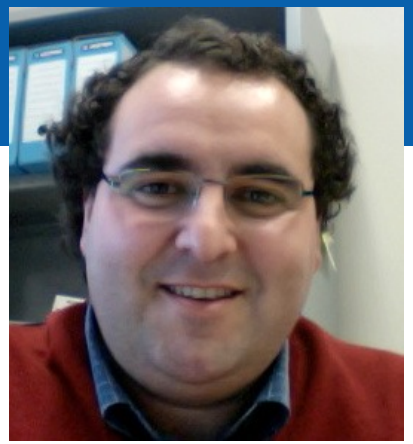
Sesión 0
Organización e introducción

Antonio Moratilla Ocaña

WHOAMI



- **Dr. D. Antonio Moratilla Ocaña**
  Departamento de Ciencias de la Computación
  Profesor del área de Lenguajes y Sistemas
  - Co-Director de la Cátedra de Planificando de Sistemas Inteligentes para la Optimización de la Distribución y el Transporte.
  - Co-Director de la Cátedra Narrativa en Inteligencia Artificial y Generación de Data-to-Text.
- Comunicaciones de la asignatura:
  Opción "Mensajes del curso" de la BlackBoard
- Despacho N334 - Tutorías: online cuando queráis, presencial miércoles de 17 a 21h previa solicitud y confirmación vía BlackBoard.
- Correo electrónico: Antonio.Moratilla@uah.es para cosas DISTINTAS a la asignatura (TFGs, curiosidades, dudas general, orientación profesional)
- Otras cosas (becas, cosas de interés,…): Twitter: @amoratil

Bibliografía de referencia de la asignatura

- Signatura biblioteca S004.4'4AHO
  Compiladores: principios, técnicas y herramientas
  Alfred V. Aho, Mónica S. Lam, Ravi Sethi, Jeffrey D. Ullman
- Signatura biblioteca S004.4'4LOU
  Construcción de compiladores : principios y práctica
  Kenneth C. Louden
- Signatura biblioteca S004.4'4MAR
  Teoría, diseño e implementación de compiladores de lenguajes
  Francisco Javier Martínez López, Alejandro Ramallo Martínez

Organización

Teoría

- Bloque 1:
  - Introducción
  - Análisis Léxico
  - Análisis Sintáctico
- Bloque 2:
  - Análisis Semántico
  - Herramientas
  - Procesos
  - Entornos

Laboratorio

- Bloque 1:
  - Expresiones Regulares
  - Autómatas
  - Gramáticas
- Bloque 2:
  - Gramáticas
  - Semántica
  - Soluciones complejas

Organización

- Clase de teoría
  - Exposición: de 30 minutos a 1h.
  - Ejercicios: de 30 minutos a 1h.
  - Prueba competitiva: máximo 15 minutos.
- Evaluación teórica:
  - 2 pruebas (exámenes) de teoría + práctica.
  - Bonus en cada prueba conseguidos con las pruebas competitivas (+0,5 puntos por sesión a los 3 mejores de cada sesión): **kahoot.it**

Procesadores de ¿Lenguajes?

- Lenguajes:
  - Naturales (No Formales)
    - Español, inglés, …
  - Formales
    - Lenguajes de programación, …

- Un lenguaje formal es un lenguaje cuyos símbolos primitivos y reglas para unir esos símbolos están formalmente especificados.

¿Por qué hay lenguajes de programación?

- **Lenguajes Máquina**
  - Son los lenguajes de más bajo nivel: secuencias binarias de ceros y unos.
  - Históricamente, los primeros
- **Lenguajes Ensambladores**
  - Segunda generación de lenguajes
  - Versión simbólica de los lenguajes máquina (MOV, ADD, etc).
- **Lenguajes de Alto Nivel**
  - Lenguajes de tercera generación (3GL)
    - Estructuras de control, Variables de tipo, Recursividad, etc.
    - Ej.: C, Pascal, C++, Java, etc
- **Lenguajes Orientados a Problemas: describen la solución, no cómo conseguirla**
  - Lenguajes de cuarta generación (4GL) Ej. SQL

¿Necesitamos más lenguajes?

- Los lenguajes de programación representan un conjunto de construcciones abstractas centradas en resolver problemas más o menos genéricos, en base a una serie de paradigmas de organización del pensamiento y su desarrollo.

- Si **los problemas evolucionan**, es natural que los lenguajes para resolverlos también, e incluso aparezcan **nuevos lenguajes**.

- ELM, DART, TypeScript, Go, Babel, Kotlin, Rust, Crystal, Elixir,… son lenguajes que en enero de 2010 no existían.

Traductores, intérpretes y compiladores

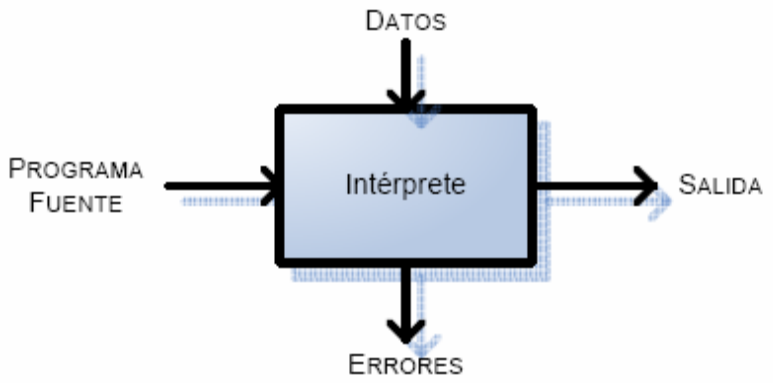
- **Traductor:** Un traductor es un programa que lee un programa escrito en lenguaje **fuente** de alto nivel, y lo traduce a un lenguaje **objeto**.

- Según sus funcionalidades puede ser de dos tipos:
  - **Intérprete**: Ejecuta directamente lo que traduce, sin almacenar en disco la traducción realizada.
  - **Compilador**: Genera un fichero del lenguaje objeto estipulado, que puede posteriormente ser ejecutado tantas veces se necesite sin volver a traducir.
    - Como beneficio adicional, el compilador informa de los errores del programa fuente, ya que lo analiza al completo.

Tipos de traductor: INTÉRPRETE

El intérprete ejecuta el código según lo va leyendo.

1. Cada vez que se escribe una línea el programa comprueba si es correcta, si lo es, la ejecuta.
2. La ejecución es interactiva.
3. Los intérpretes más puros no guardan copia del programa que se está escribiendo



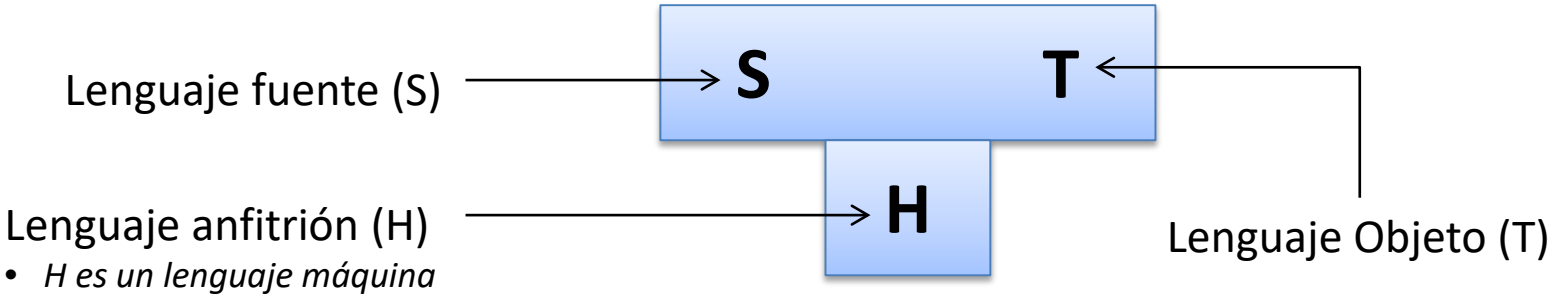
– Ejemplos: Procesos por lotes, JavaScript, CAML, etc.
– El intérprete siempre debe estar presente para la traducción y ejecución.

Tipos de traductor: COMPILADOR

- Un compilador es un programa que lee un programa escrito en lenguaje fuente, y lo traduce a un lenguaje objeto de bajo nivel. Además generará una lista de los posibles errores que tenga el programa fuente.
  - El compilador es el traductor más extendido
  - El programa ejecutable, una vez creado, **no necesita el compilador** para funcionar.

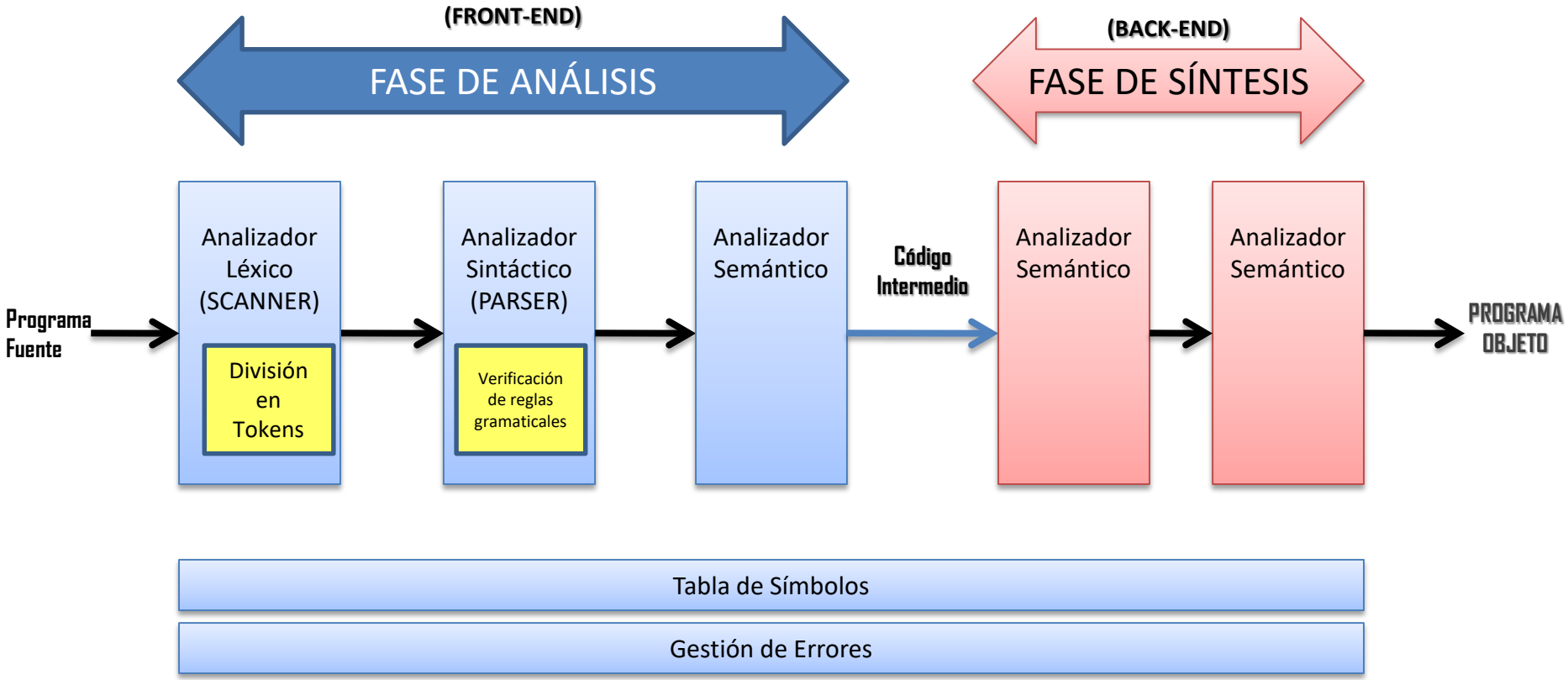
Representación de un compilador

Los compiladores pueden estar implementados en lenguajes de programación distintos del lenguaje fuente (**S**ource) y del lenguaje objeto (**T**arget). A este tercer lenguaje se le denomina lenguaje anfitrión (**H**ost). Por tanto, para representar en abstracto un compilador suele emplearse una representación en forma de **T** que incluye los 3 lenguajes que lo caracterizan



- *H es un lenguaje máquina frecuentemente igual a T*

Procesamiento de un lenguaje



Fuentes

- Para la elaboración de estas transparencias se han utilizado:
  - Transparencias de cursos previos (elaboradas por los profesores Dr. D. Salvador Sánchez, Dr. D. José Luis Cuadrado).
  - Libros de referencia.