La semántica operacional, en el contexto de los lenguajes de programación, describe el significado de un programa como una serie de pasos computacionales que simulan la ejecución de ese programa en una máquina abstracta. Se enfoca en cómo el lenguaje se ejecuta, especificando la secuencia de estados que el programa atraviesa durante su ejecución.

El significado de la semántica operacional:

- **Definición formal:**
  La semántica operacional define el significado de un lenguaje de programación al describir cómo una construcción del lenguaje se ejecuta.
-

Ejemplo sencillo:
Imagina un programa que suma dos números. La semántica operacional podría definir los siguientes pasos:

1. **Estado inicial:** El programa recibe dos valores (a y b) y un operador (+).
2. **Paso 1:** Se realiza la suma a + b.
3. **Estado final:** El resultado de la suma es el valor del programa.

La semántica operacional es una herramienta fundamental para entender cómo un lenguaje de programación funciona en la práctica y es utilizada para verificar la corrección de compiladores y otros procesadores de lenguajes
Simulación de la ejecución:
Funciona como una simulación de la máquina abstracta que "ejecuta" el código.
Estados y transiciones:
La ejecución se modela como una serie de transiciones de estado, donde cada estado representa la situación del programa en un punto determinado de su ejecución.
Foco en la ejecución:
A diferencia de otras semánticas (como la semántica denotacional), la semántica operacional no se centra en el significado conceptual abstracto del programa, sino en su comportamiento durante la ejecución.
Distintas variantes:
Existen diferentes tipos de semántica operacional, como la de paso pequeño y la de paso grande, que varían en la granularidad de los pasos de ejecución.

En semántica operacional, la "ligadura" (binding) es la asociación entre un nombre (identificador) y un valor o una entidad (variable, función, tipo, etc.). Esta vinculación define cómo se interpretan las expresiones que usan el nombre, permitiendo acceder a la información asociada.

Concepto general:
La ligadura establece una relación entre un nombre y su significado. En otras palabras, indica qué objeto o valor representa ese nombre dentro de un contexto específico.

Ejemplos:

- En la declaración de una variable, la ligadura vincula el nombre de la variable (identificador) con una ubicación de memoria que contiene el valor.

En la llamada de una función, la ligadura vincula el nombre de la función con su código.

En la definición de un tipo, la ligadura vincula el nombre del tipo con la estructura y propiedades que define

Significado en semántica operacional:
La semántica operacional describe cómo un programa se ejecuta como una secuencia de pasos. La ligadura es crucial en este proceso, ya que define cómo los nombres son utilizados para acceder a los valores y realizar operaciones en la máquina abstracta que define la semántica operacional.

Diferentes tipos de ligadura:
Se pueden distinguir diferentes tipos de ligadura según cuándo y cómo se establecen:

- **Estática (binding time):** La ligadura se establece durante la compilación o traducción del código.

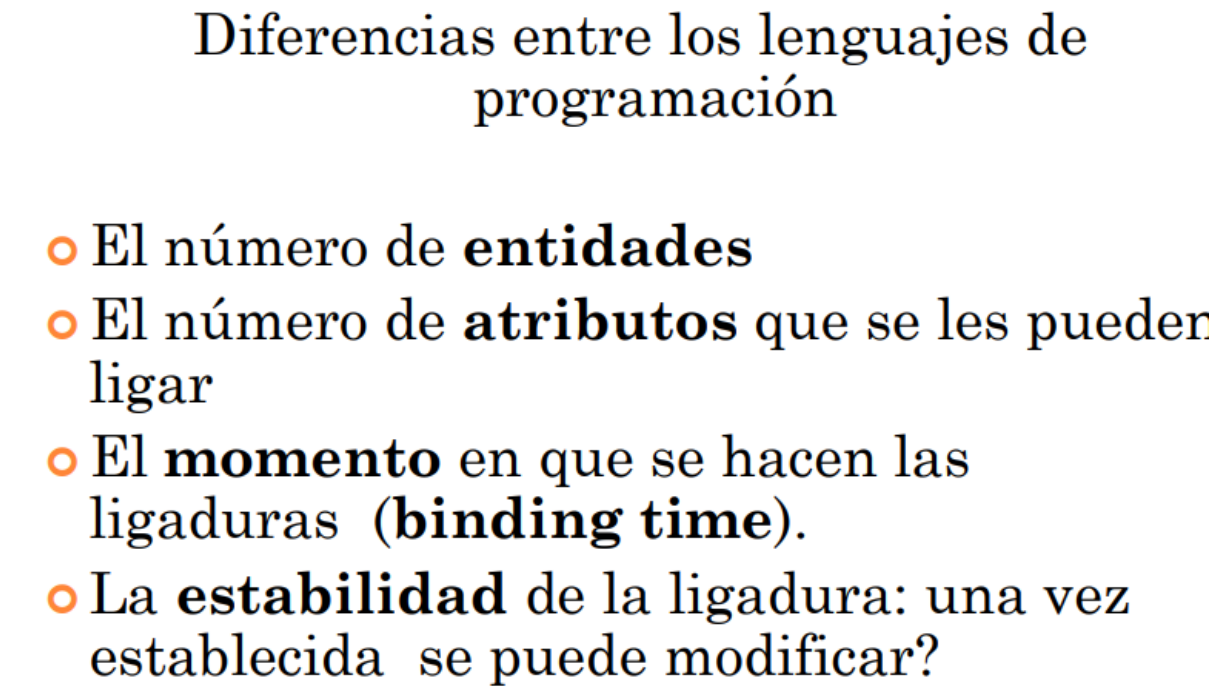
Dinámica (binding time): La ligadura se establece durante la ejecución del programa.

Estabilidad: Se refiere a si la ligadura puede cambiar durante la ejecución (ligadura dinámica) o si permanece fija (ligadura estática).

-

Importancia:
La ligadura es fundamental para la correcta interpretación y ejecución de los programas, ya que define cómo los nombres son usados para acceder a los valores y realizar operaciones.

Relación con otros conceptos:
La ligadura está estrechamente relacionada con otros conceptos de los lenguajes de programación, como el alcance (scope) y la resolución de nombres (name resolution), que determinan dónde y cómo se pueden utilizar los nombres dentro de un programa.



En Python, el concepto de "momento de ligadura" (binding time) se refiere a cuándo se resuelve la relación entre un nombre de variable y el objeto al que se refiere. En Python, esta ligadura es dinámica, lo que significa que se realiza en tiempo de ejecución (runtime), no en tiempo de compilación.
Explicación:

- **Ligadura estática:**
  En lenguajes de programación estáticos (como C++), la ligadura se realiza en tiempo de compilación. Esto significa que el tipo de una variable y la dirección de memoria a la que se refiere se determinan antes de que el programa se ejecute.
- **Ligadura dinámica:**
  En Python, que es un lenguaje de programación dinámico, la ligadura es dinámica. Esto significa que la variable puede referirse a diferentes tipos de objetos durante la ejecución del programa. La relación entre el nombre de la variable y el objeto se establece en tiempo de ejecución

Python

```
mi_variable = 10  # mi_variable se une al entero 10
mi variable = "Hola"  # mi variable se une a la cadena de texto "Hola"
```

Implicaciones de la ligadura dinámica:

- **Flexibilidad:**
  La ligadura dinámica permite a los programadores escribir código más flexible y dinámico, ya que pueden cambiar el tipo de objeto al que se refiere una variable durante la ejecución.
- **Coste computacional:**
  La ligadura dinámica puede implicar un coste computacional, ya que la resolución de nombres y la asignación de memoria se realizan en tiempo de ejecución.
- **Error en tiempo de ejecución:**
  Debido a que los tipos de variable no se verifican en tiempo de compilación, es posible que los errores de tipo se detecten en tiempo de ejecución.

La estabilidad de la ligadura:

- **Dinámica:**
  La ligadura dinámica permite una mayor flexibilidad, ya que las variables no están limitadas a un único tipo.
- **Implicaciones:**
  La flexibilidad de la ligadura dinámica puede hacer que el código sea más fácil de escribir y leer, pero también puede hacer que sea más difícil de depurar si no se controla cuidadosamente.
-

O sea Python, el momento de ligadura es dinámico, lo que significa que la asociación entre un nombre y un valor (o tipo) ocurre durante la ejecución. La estabilidad de la ligadura también es dinámica, lo que implica que la asociación puede ser modificada durante la ejecución.

<NOMBRE, **ALCANCE** ,TIPO, L-VALUE, R-VALUE>

Alcance estático

- Llamado **alcance léxico**.
- Define el alcance en términos de la estructura léxica del programa.
- Puede ligarse estáticamente a una declaración (explícita o implícita) examinando el texto del programa, sin necesidad de ejecutarlo.
- La mayoría de los lenguajes adoptan reglas de ligadura de alcance estático.

Alcance dinámico

- Define el alcance del nombre de la variable en términos de la ejecución del programa.
- Cada declaración de variable extiende su efecto sobre todas las instrucciones ejecutadas posteriormente, hasta que una nueva declaración para una variable con el mismo nombre es encontrado durante la ejecución.
- **APL**, **Lisp** (original), **Afnix** (llamado *Aleph* hasta el 2003), **Tcl** (Tool Command Language), **Perl**

Conceptos asociados con el alcance

- **Local:** Son todas la referencias que se han creado dentro del programa o subprograma.

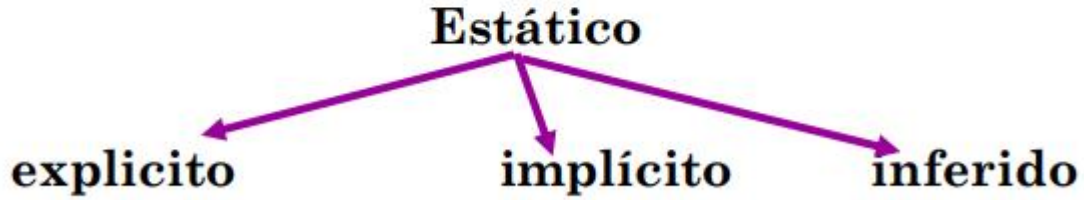
- **No Local:** Son todas las referencias que se utilizan dentro del subprograma pero que no han sido creadas en él.

- **Global:** Son todas las referencias creadas en el programa principal

<NOMBRE, ALCANCE, **TIPO**, L-VALUE, R-VALUE>

Momentos - **Estático**

- El tipo se **liga en compilación y no puede ser cambiado**
  - El chequeo de tipo también será estático



Pascal, Algol, Simula, ADA, C, C++, Java, etc

- **Área de memoria ligada a la variable**
- **Tiempo de vida** (lifetime) o extensión:

Periodo de tiempo que existe la ligadura

- **Alocación**:

Momento que se reservar la memoria

El tiempo de vida es el tiempo en que la variable esté alocada en memoria

<NOMBRE, **ALCANCE** ,TIPO, L-VALUE, R-VALUE>

- **Alcance estático**
  - Llamado **alcance léxico**.
  - Define el alcance en términos de la estructura léxica del programa.
  - Puede ligarse estáticamente a una declaración (explícita o implícita) examinando el texto del programa, sin necesidad de ejecutarlo.
  - La mayoría de los lenguajes adoptan reglas de ligadura de alcance estático.
- **Alcance dinámico**
  - Define el alcance del nombre de la variable en términos de la ejecución del programa.
  - Cada declaración de variable extiende su efecto sobre todas las instrucciones ejecutadas posteriormente, hasta que una nueva declaración para una variable con el mismo nombre es encontrado durante la ejecución.
  - **APL**, **Lisp** (original), **Afnix** (llamado *Aleph* hasta el 2003), **Tcl** (Tool Command Language), **Perl**

Momentos - **Alocación**

- **Estática:** sensible a la historia
- **Dinámica**
  - Automática; cuando aparece la declaración
  - Explícita: a través de algún constructor

- **Persistente**: su tiempo de vida no depende de la ejecución:

  existe en el ambiente

  - Archivos - Bases de datos

Estructura de ejecución de los lenguajes de programación

- **Estático**
- **Basado en pila**
- **Dinámico**

ESTATICO: ESPACIO FIJO

- El espacio necesario para la ejecución se deduce del código
- Todo los requerimientos de memoria necesarios se conocen antes de la ejecución
- La alocación puede hacerse estáticamente
- No puede haber recursión

BASADO EN PILA: ESPACIO PREDECIBLE

- El espacio se deduce del código. Algol-60
- Programas más potentes cuyos requerimientos de memoria no puede calcularse en traducción.
- La memoria a utilizarse es **predecible** y sigue una disciplina last-in-first-out.
- Las variables se alocan automáticamente y se desalocan cuando el alcance se termina
- Se utiliza una estructura de pila para modelizarlo.