UNIVERSIDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE
INFORMÁTICA Y ANÁLISIS NUMÉRICO

LENGUAJES
DE INTELIGENCIA ARTIFICIAL

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

SEGUNDO CURSO

PRIMER CUATRIMESTRE

CURSO ACADÉMICO 2009 - 2010

Primera parte: Scheme

- Tema 1.- Introducción al Lenguaje Scheme
- Tema 2.- Expresiones y Funciones
- Tema 3.- Predicados y sentencias condicionales
- Tema 4.- Iteración y Recursión
- Tema 5.- Tipos de Datos Compuestos
- Tema 6.- Abstracción de Datos
- Tema 7.- Lectura y Escritura

Segunda parte: Prolog

- Tema 8.- Introducción al Lenguaje Prolog
- Tema 9.- Elementos Básicos de Prolog
- Tema 10.- Listas
- Tema 11.- Reevaluación y el "corte"
- Tema 12.- Entrada y Salida

Primera parte: Scheme

- Tema 1.- Introducción al Lenguaje Scheme
- Tema 2.- Expresiones y Funciones
- Tema 3.- Predicados y sentencias condicionales
- Tema 4.- Iteración y Recursión
- Tema 5.- Tipos de Datos Compuestos
- Tema 6.- Abstracción de Datos
- Tema 7.- Lectura y Escritura

Índice

1. Características Fundamentales de la Programación Funcional
2. Reseña Histórica de Scheme

Índice

1. **Características Fundamentales de la Programación Funcional**
2. Reseña Histórica de Scheme

1. **Características Fundamentales de la Programación Funcional**
   - ✓ La Programación **Funcional** es un **subtipo** de la Programación **Declarativa**

1. **Características Fundamentales de la Programación Funcional**
   - ✓ **Programación Declarativa (1 / 2)**
     - ➢ **Objetivo**: **Descripción del Problema**

"**Qué**" problema debe ser resuelto?

- **Observación**:
  - No importa "**cómo**" es resuelto el problema
  - **Evita** aspectos o características de implementación

1. **Características Fundamentales de la Programación Funcional**
   - ✓ **Programación Declarativa (2 / 2)**
     - ➢ **Características**
       - Expresividad
       - Extensible: regla del 10% - 90%
       - Protección
       - Elegancia Matemática
     - ➢ **Tipos**:
       - **Programación Funcional o Aplicativa**:
         - Lisp, **Scheme**, Haskell, …
       - Programación **Lógica**: **Prolog**

1. **Características Fundamentales de la Programación Funcional**
   - ✓ **Principio** de la Programación **Funcional "Pura"**

"El **valor de la expresión** sólo **depende** del valor de sus **subexpresiones**, si las tiene"

- ✓ **No existen efectos colaterales**

El valor de "a + b" **sólo** depende de "a" y "b".

- ✓ El término **función** es usado en su sentido **matemático**
- ✓ **No hay instrucciones**: programación **sin** asignaciones
  - ➢ La Programación Funcional **impura permite** la

"sentencia de asignación"

1. **Características Fundamentales de la Programación Funcional**
   - ✓ **Estructura de los programas** *en la Programación* **Funcional**
     - ➢ **El programa es una función compuesta de Funciones más simples**
     - ➢ **Ejecución de una Función:**
       - **Recibe los datos de entrada:** *argumentos o parámetros de las funciones*
       - **Evalúa las expresiones**
       - **Devuelve el resultado**: *valor calculado por la función*

1. **Características Fundamentales de la Programación Funcional**
   - ✓ **Tipos de lenguajes funcionales**
     - ➢ *La* **mayoría** *son lenguajes* **interpretados**
     - ➢ *Algunas versiones son lenguajes* **compilados**
   - ✓ **Gestión de la memoria**
     - ➢ **Gestión implícita de la memoria**:
       - *La gestión de la memoria es una* **tarea** *del* **intérprete**.
       - *El programador* **no** *debe* **preocuparse** *por la gestión de la memoria.*
     - ➢ **Recolección de basura:** *tarea del intérprete.*

En resumen: *el programador sólo se debe de preocupar por la* **Descripción del Problema**

Índice

2. **Reseña Histórica de Scheme**
   - ✓ LISP
   - ✓ Comparación entre Compilación e Interpretación
   - ✓ Comparación entre el ámbito léxico (o estático) y el dinámico
   - ✓ Origen de Scheme

2. **Reseña Histórica de Scheme**
   - ✓ **LISP**
   - ✓ Comparación entre Compilación e Interpretación
   - ✓ Comparación entre el ámbito léxico (o estático) y el dinámico
   - ✓ Origen de Scheme

2. **Reseña Histórica de Scheme**
   - ✓ **LISP**
     - ➢ **John McCarthy** (MIT)
     - ➢ El programa **"Advice Taker"**:
       - Fundamentos teóricos: Lógica Matemática
       - Objetivo: Deducción e inferencias
     - ➢ **LISP**: **LIS**t **P**rocessing (1956 – 1958)
       - Segundo lenguaje histórico de **Inteligencia Artificial** (después de IPL)
       - En la actualidad, segundo lenguaje histórico **en uso** (después de Fortran)
       - LISP está basado en el Lambda Calculus (**Alonzo Church**)
     - ➢ **Scheme** es un **dialecto** de **LISP**

2. **Reseña Histórica de Scheme**
   - ✓ **LISP**
     - ➢ **Características de la Programación Funcional**
       - **Recursión**
       - **Listas**
       - Gestión **implícita** de la memoria
       - Programas interactivos e **interpretados**
       - Programación **Simbólica**
       - Reglas de ámbito **Dinámico** para identificadores **no** locales

16

2. **Reseña Histórica de Scheme**
   - ✓ **LISP**
     - ➢ **Contribuciones de** LISP:
       - Funciones **"Built – in"**
       - **Recolección de basura**
       - **Lenguaje de Definición Formal**: el propio lenguaje **LISP**

17

2. **Reseña Histórica de Scheme**
   - ✓ **LISP**
     - ➢ **Aplicaciones**: Programas de **Inteligencia Artificial**
       - Verificación y prueba de Teoremas
       - Diferenciación e Integración Simbólica
       - Problemas de Búsqueda
       - Procesamiento del Lenguaje Natural
       - Visión Artificial
       - Robótica
       - Sistemas de Representación del Conocimiento
       - Sistemas Expertos
       - Etc.

18

2. **Reseña Histórica de Scheme**
   - ✓ **LISP**
     - ➢ **Dialectos (1 /2)**
       - **Mac LISP** (Man and computer or Machine – aided cognition): Versión de la Costa **Este**
       - **Inter LISP** (Interactive LISP): Versión de la Costa **Oeste**
         - Compañía de Bolt, Beranek y Newman (BBN)
         - Centro de Investigación de Xerox en Palo Alto (Texas)
         - **Máquina LISP**

2. **Reseña Histórica de Scheme**
   - ✓ **LISP**
     - ➢ **Dialectos** (2 / 2)
       - **Mac LISP** (Man and computer or Machine – aided cognition): East Coast Version
         - C-LISP: Universidad de Massachusetts
         - Franz LISP: Universidad de California (Berkeley). **Versión compilada**.
         - NIL (New implementation of LISP): MIT.
         - PSL (Portable Standard LISP): Universidad de Utah
         - **Scheme**: MIT.
         - T (True): Universidad de Yale
         - Common LISP

2. **Reseña Histórica de Scheme**
   - ✓ LISP
   - ✓ **Comparación entre Compilación e Interpretación**
   - ✓ Comparación entre el ámbito léxico (o estático) y el dinámico
   - ✓ Origen de Scheme

2. Reseña Histórica de Scheme

✓ Comparación entre Compilación e Interpretación

➢ Compilación:

- *El código fuente (**alto nivel**) es transformado en código ejecutable (**bajo nivel**), que puede ser ejecutado independientemente.*

2. Reseña Histórica de Scheme

✓ Comparación entre Compilación e Interpretación

➢ Compilación

Código fuente → Compilador

2. Reseña Histórica de Scheme

✓ Comparación entre Compilación e Interpretación

➢ Compilación

Código fuente → Compilador

↓

Errores de compilación

2. Reseña Histórica de Scheme
   - ✓ Comparación entre Compilación e Interpretación
     - ➢ Compilación

25

2. Reseña Histórica de Scheme
   - ✓ Comparación entre Compilación e Interpretación
     - ➢ Compilación

26

2. Reseña Histórica de Scheme
   - ✓ Comparación entre Compilación e Interpretación
     - ➢ Compilación

27

2. Reseña Histórica de Scheme
   - ✓ Comparación entre Compilación e Interpretación
     - ➢ Compilación

28

2. Reseña Histórica de Scheme
   - ✓ Comparación entre Compilación e Interpretación
     - ➢ Interpretación

29

2. Reseña Histórica de Scheme
   - ✓ Comparación entre Compilación e Interpretación
     - ➢ **Interpretación** o **simulación**: consiste en un ciclo de tres fases

30

2. **Reseña Histórica de Scheme**

- ✓ **Comparación entre Compilación e Interpretación**
  - ➢ **Interpretación** o simulación: consiste en un ciclo of tres fase
    1. **Análisis**: el código fuente es analizado para determinar la siguiente sentencia correcta que va a ser ejecutada.

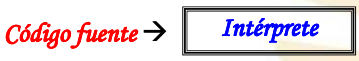
2. **Reseña Histórica de Scheme**

- ✓ **Comparación entre Compilación e Interpretación**
  - ➢ **Interpretación** o simulación: consiste en un ciclo of tres fase
    1. **Análisis**: el código fuente es analizado para determinar la siguiente sentencia correcta que va a ser ejecutada.
    2. **Generación:** la sentencia es transformada en Código ejecutable.

2. **Reseña Histórica de Scheme**

- ✓ **Comparación entre Compilación e Interpretación**
  - ➢ **Interpretación** o simulación: consiste en un ciclo of tres fase
    1. **Análisis**: el código fuente es analizado para determinar la siguiente sentencia correcta que va a ser ejecutada.
    2. **Generación:** la sentencia es transformada en Código ejecutable.
    3. **Ejecución**: el código generado es ejecutado.

2. Reseña Histórica de Scheme
   - ✓ Comparación entre Compilación e Interpretación
     - ➢ Interpretación

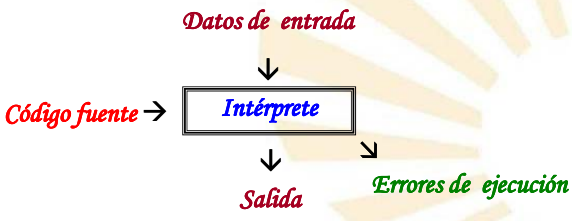


34

2. Reseña Histórica de Scheme
   - ✓ Comparación entre Compilación e Interpretación
     - ➢ Interpretación

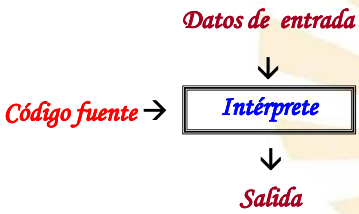
35

2. Reseña Histórica de Scheme
   - ✓ Comparación entre Compilación e Interpretación
     - ➢ Interpretación



36

2. Reseña Histórica de Scheme
   - ✓ Comparación entre Compilación e Interpretación
     - ➢ Interpretación



2. Reseña Histórica de Scheme
   - ✓ Comparación entre Compilación e Interpretación

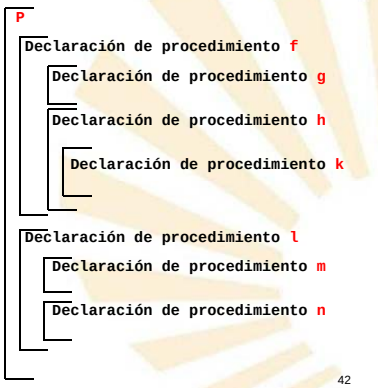
Compilación	Interpretación
- Independencia	- Dependencia
- Necesidades de memoria	- Sin necesidades de memoria
- Eficiencia	- Menos eficiencia
- Global	- Local
- No interactividad	- Interactividad
- Código cerrado durante la ejecución	- Código abierto durante la ejecución

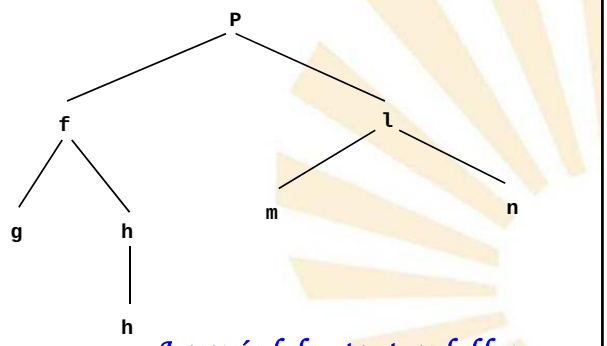
2. Reseña Histórica de Scheme
   - ✓ LISP
   - ✓ Comparación entre Compilación e Interpretación
   - ✓ Comparación entre el ámbito léxico (o estático) y el dinámico
   - ✓ Origen de Scheme

2. **Reseña Histórica de Scheme**
   - ✓ **Comparación entre el ámbito léxico (o estático) y el dinámico**
     - Las **reglas** de **ámbito** determinan la **declaración** de identificadores **no** locales
     - Identificadores **no** locales:
       - **Variables, Funciones o Procedimientos** que pueden ser usados en un función o procedimiento pero que **no están declarados** in esa función o procedimiento
     - **Dos tipos**
       - **Ámbito Léxico o Estático**
         - **Con** "estructura de bloques": Pascal, **Scheme**
         - **Sin** "estructura de bloques": C, Fortran
       - **Ámbito Dinámico**
         - **Siempre con** "estructura de bloques": Lisp, SNOBOL, APL

2. **Reseña Histórica de Scheme**
   - ✓ **Comparación entre el ámbito léxico (o estático) y el dinámico**
     - **Estructura de bloques**
       - Un procedimiento o función puede **llamar** a
         - Sí mismo
         - Sus hijos (pero **no** a sus nietos…)
         - Sus hermanos (pero **no** a sus sobrinos)
         - Su padre, abuelo, bisabuelo, …
         - Los hermanos de su padres, abuelo, …
       - Un procedimiento o función puede **ser llamado** por
         - Sí mismo
         - Su padre (pero **no** por su abuelo, …)
         - Sus hijos, nietos, bisnietos, …
         - Sus hermanos y sus hijos, nietos, ...

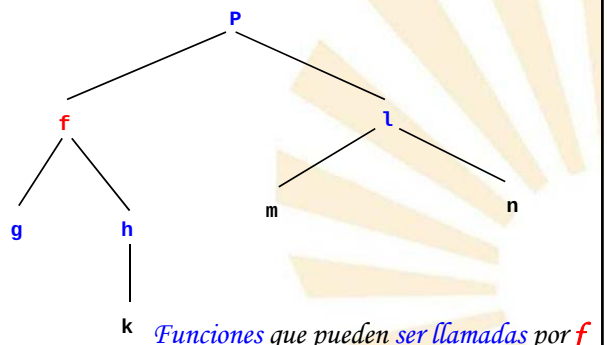
Ejemplo de estructura de bloques





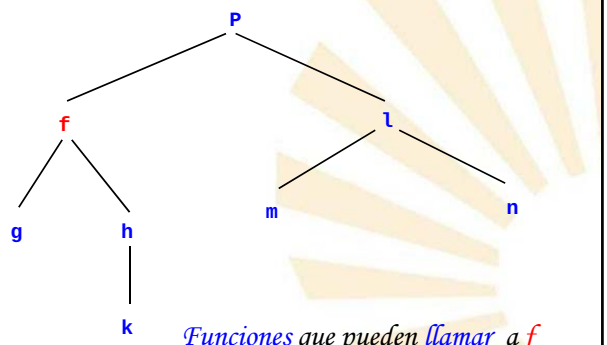
Jerarquía de la estructura de bloques

43



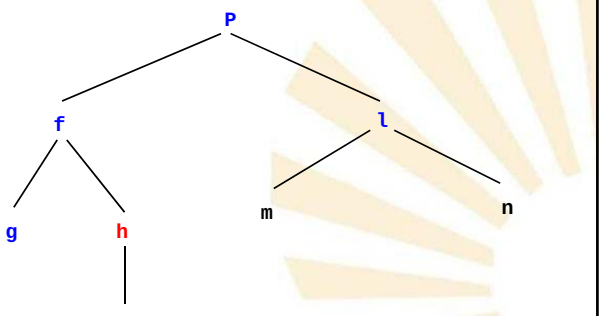
Funciones que pueden ser llamadas por f

44

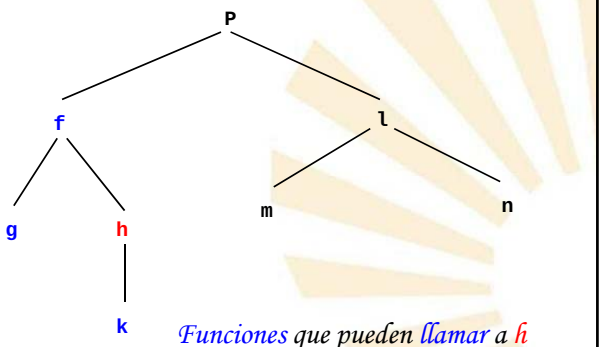


Funciones que pueden llamar a f

45



Funciones que pueden ser llamadas por h



Funciones que pueden llamar a h

2. **Reseña Histórica de Scheme**
   - ✓ **Comparación entre el ámbito léxico (o estático) y el dinámico**
     - ➢ **Ámbito léxico o estático**
       - *La* ***declaración*** *de un identificador* ***no*** *local depende del* ***contexto léxico más cercano:***

         Sólo tienes que ***leer*** *el programa*

         para determinar la declaración de un identificador
       - ***Reglas del anidamiento más cercano***
         - *El* ***ámbito*** *de un procedimiento (*)* ***f*** *incluye al procedimiento* ***f****.*
         - *Si un identificador* ***no*** *local* ***x*** *es usado en* ***f*** *entonces la declaración de* ***x*** *debe ser encontrada en procedimiento más* ***cercano g*** *que incluya a* ***f***
         - ***Observación*** *(*): procedimiento, función o bloque.*

Ejemplo:

Ámbito léxico

con

"estructura de bloques"

```
Declaración de procedimiento h
 Declaración de una variable x  (x1)
 Declaración de una variable y  (y1)
 Declaración de una variable z  (z1)

 Declaración de procedimiento g
  Declaración de una variable x (x2)
  Declaración de una variable y (y2)

  Declaración de procedimiento f
    Declaración de una variable x (x3)

    Uso de x  (→ x3)
    Uso de y  (→ y2)
    Uso de z  (→ z1)

    Uso de x  (→ x2)
    Uso de y  (→ y2)
    Uso de z  (→ z1)
    Llamada a f

    Uso de x  (→ x1)
    Uso de y  (→ y1)
    Uso de z  (→ z1)
    Llamada a g
```


2. *Reseña Histórica de Scheme*
   - ✓ *Comparación entre el ámbito léxico (o estático) y el dinámico*
     - ➢ *Ámbito léxico o estático*
       - ▪ ***Sin*** *estructura de bloques*
         - *Si* ***x*** ***no*** *es local para una función* ***especifica*** *entonces* ***no*** *es local para* ***todas*** *las funciones*

Ejemplo en C:

Ámbito léxico

sin

"estructura de bloques"

```
int x;  /* x1 */
int y;  /* y1 */
int z;  /* z1 */

main()
        {
   int x;  /* x2 */
   int y;  /* y2 */

   /* Uso de x → x2 */
   /* Uso de y → y2 */
   /* Uso de z → z1 */
   /* Llamada a f */
   f ();
  }

 f()
  {
   int x;  /* x3 */
   /* Uso de x → x3 */
   /* Uso de y → y1 */
   /* Uso de z → z1 */
      }
```

Las variables globales no son recomendables

2. **Reseña Histórica de Scheme**
   - ✓ **Comparación entre el ámbito léxico (o estático) y el dinámico**
     - ➢ **Ámbito dinámico:**
       - La **declaración** de un **identificador** depende de la **ejecución del programa**

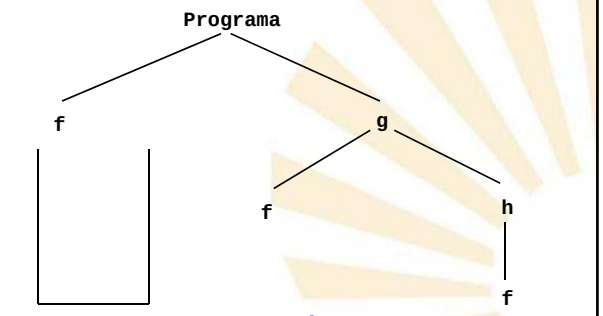
         Tienes que **ejecutar** el programa

         para determinar la declaración de un identificador
       - **Reglas de la activación más cercana**
         - El **ámbito** de un procedimiento (*) **f** incluye al procedimiento **f**.
         - Si un identificador **no** local **x** es usado en la **activación** de **f** entonces la declaración de **x** debe ser encontrada en el procedimiento **activo más cercano g** con una declaración de x.
         - **Observación** (*): procedimiento, función o bloque

2. **Reseña Histórica de Scheme**
   - ✓ **Comparación entre el ámbito léxico (o estático) y el dinámico**
     - ➢ **Observación:**
       - El **Ámbito dinámico** permite que un **identificador** pueda estar asociado a **declaraciones diferentes** durante la ejecución del programa.

Ejemplo: Comparación entre los ámbitos léxico y dinámico

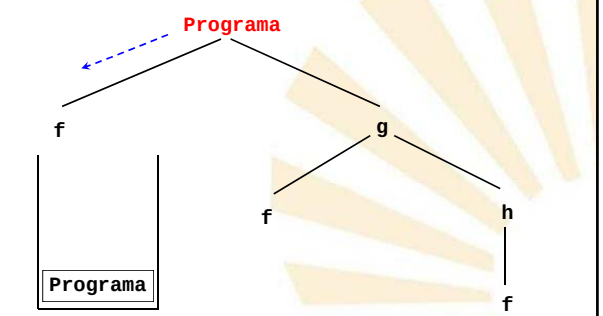
```
Programa
    Declaración de una variable x
    Declaración de procedimiento f
     Uso de x
    Declaración de procedimiento g
     Declaración de una variable x
     Declaración de procedimiento h
      Uso de x
      Llamada a f
       Llamada a f
       Llamada a h
       Si condición = verdadero
          entonces Llamada a g
          si no    Uso de x
       fin si
     Uso de x
     Llamada a f
     Llamada a g
```



Pila de Activación

Árbol de Activación

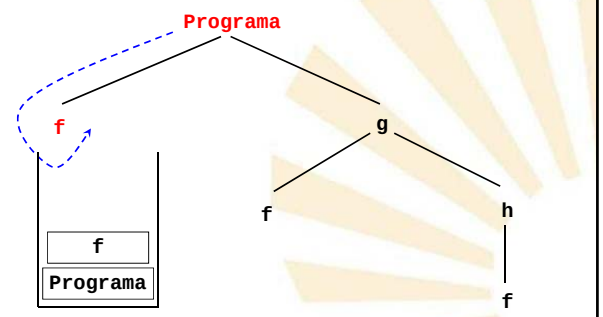
55



Pila de Activación

Árbol de Activación

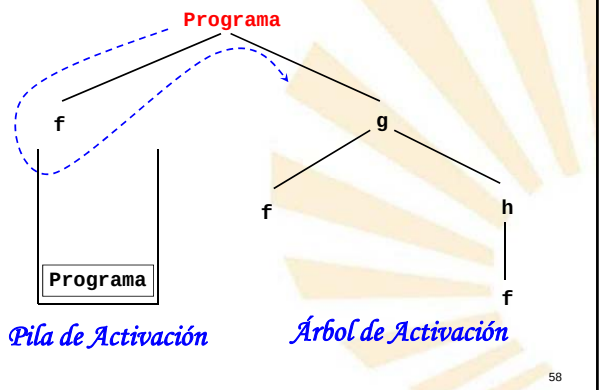
56



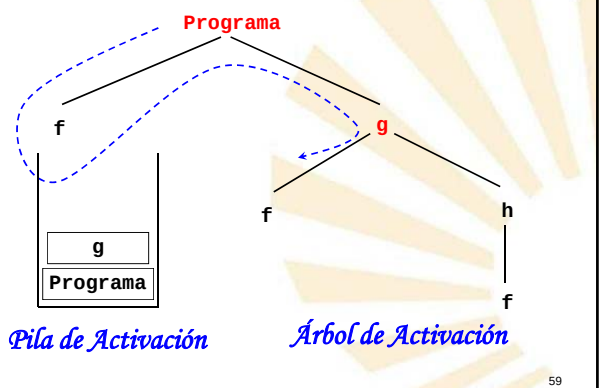
Pila de Activación

Árbol de Activación

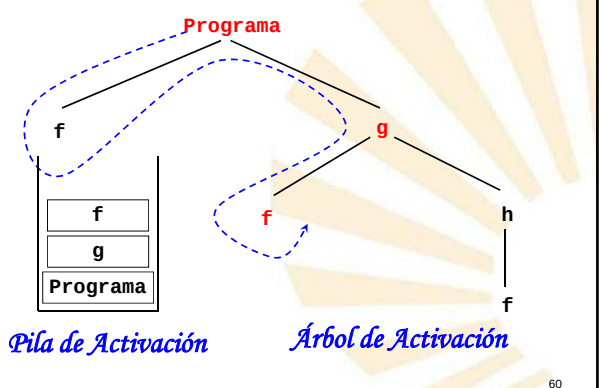
57


Programa
f
g
f
h
f
Programa
Pila de Activación
Árbol de Activación

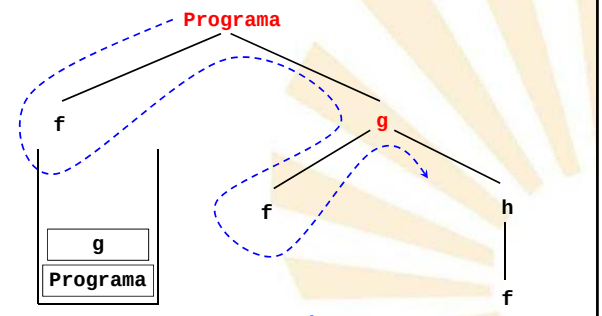
58


Programa
f
g
f
h
f
g
Programa
Pila de Activación
Árbol de Activación

59

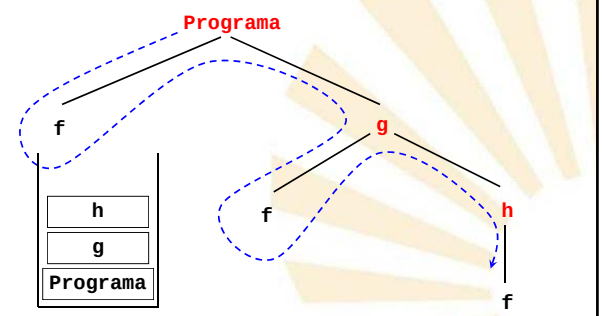

Programa
f
g
f
h
f
f
g
Programa
Pila de Activación
Árbol de Activación

60



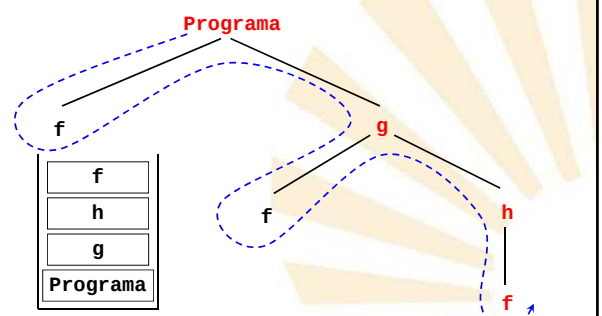
Pila de Activación — Árbol de Activación

61



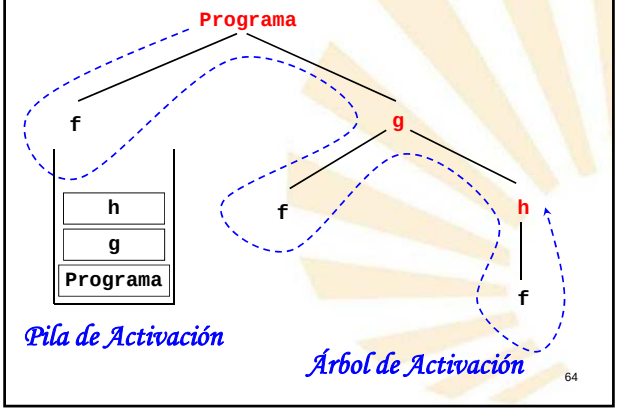
Pila de Activación — Árbol de Activación

62



Pila de Activación — Árbol de Activación

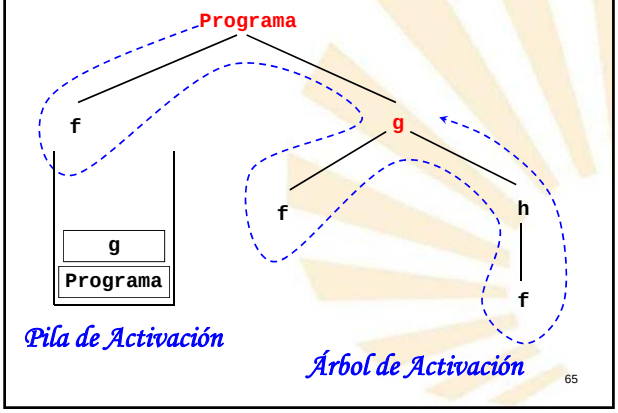
63



Pila de Activación

Árbol de Activación

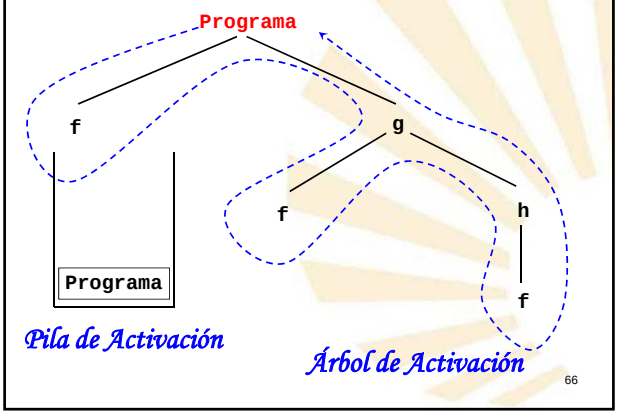
64



Pila de Activación

Árbol de Activación

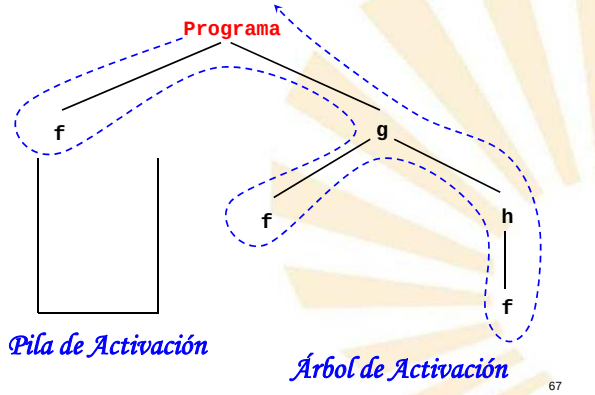
65



Pila de Activación

Árbol de Activación

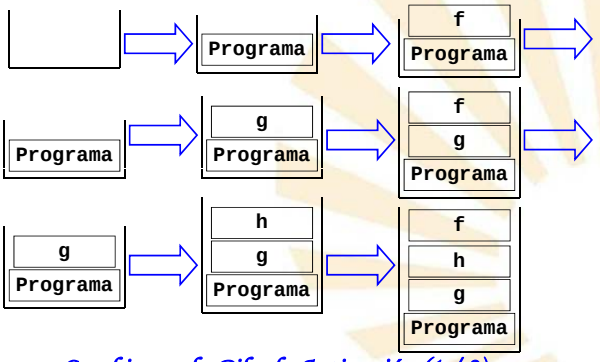
66



Pila de Activación

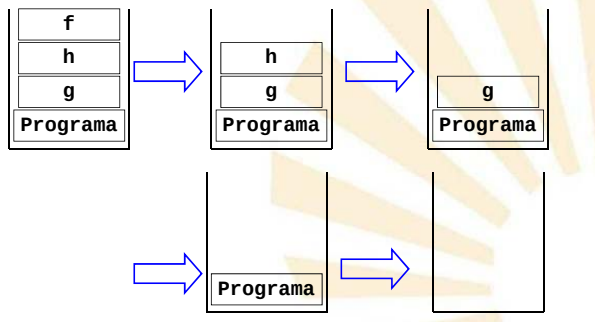
Árbol de Activación

67



Cambios en la Pila de Activación (1 / 2)

68

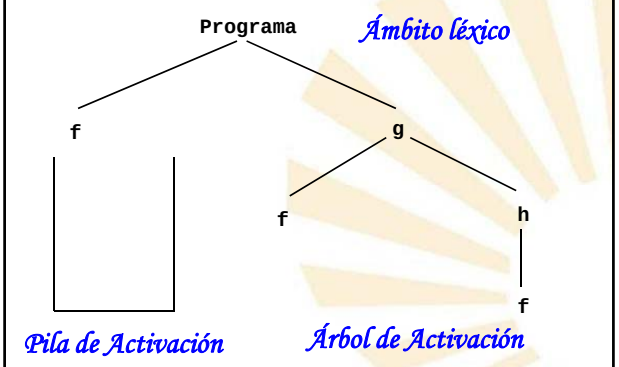


Cambios en la Pila de Activación (2 / 2)

69

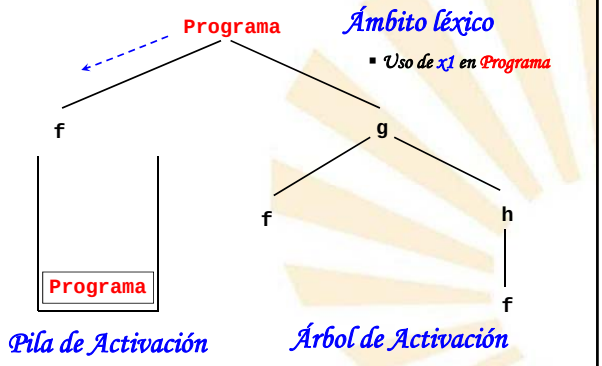
Ejecución con Ámbito léxico

```
Programa
    Declaración de una variable x  (x1)
    Declaración de procedimiento f
     Uso de x
    Declaración de procedimiento g
     Declaración de una variable x  (x2)
     Declaración de procedimiento h
      Uso de x
      Llamada a f
      Llamada a f
      Llamada a h
      Si condición = verdadero
         entonces Llamada a g
         si no    Uso de x
      fin si
     Uso de x
     Llamada a f
     Llamada a g
```

Ejecución con Ámbito léxico

```
Programa
    Declaración de una variable x  (x1)
    Declaración de procedimiento f
     Uso de x
    Declaración de procedimiento g
     Declaración de una variable x  (x2)
     Declaración de procedimiento h
      Uso de x
      Llamada a f
      Llamada a f
      Llamada a h
      Si condición = verdadero
         entonces Llamada a g
         si no    Uso de x
      fin si
     Uso de x
     Llamada a f
     Llamada a g
```



Ámbito léxico

- Uso de x1 en Programa

Pila de Activación

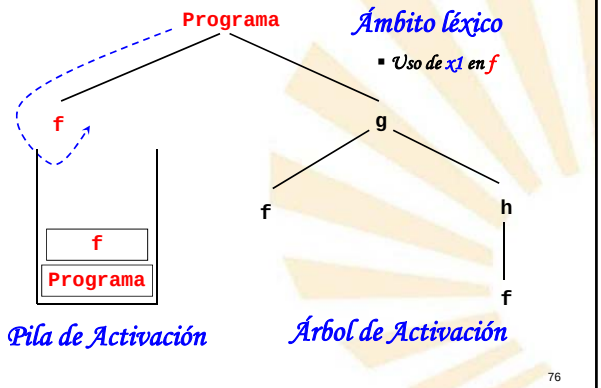
Árbol de Activación

Ejecución con Ámbito léxico

```
Programa
    Declaración de una variable x  (x1)
    Declaración de procedimiento f
     Uso de x
    Declaración de procedimiento g
     Declaración de una variable x  (x2)
     Declaración de procedimiento h
      Uso de x
      Llamada a f
       Llamada a f
       Llamada a h
       Si condición = verdadero
          entonces Llamada a g
          si no    Uso de x
       fin si
     Uso de x
     Llamada a f
     Llamada a g
```


Ejecución con Ámbito léxico

```
Programa
    Declaración de una variable x  (x1)
    Declaración de procedimiento f
     Uso de x
    Declaración de procedimiento g
     Declaración de una variable x  (x2)
     Declaración de procedimiento h
      Uso de x
      Llamada a f
       Llamada a f
       Llamada a h
       Si condición = verdadero
          entonces Llamada a g
          si no    Uso de x
       fin si
     Uso de x
     Llamada a f
     Llamada a g
```

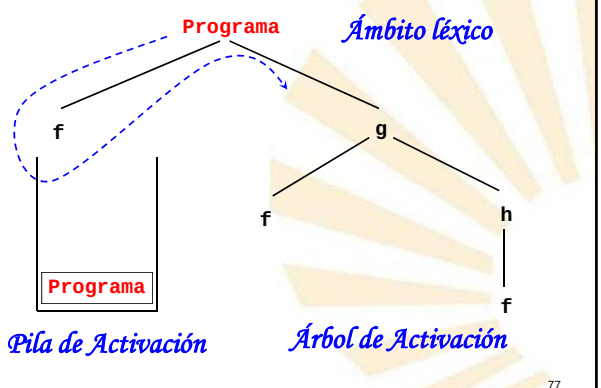


Ámbito léxico

- Uso de x1 en f

Pila de Activación

Árbol de Activación



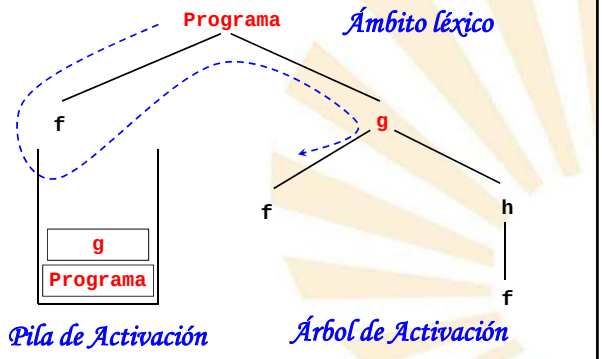
Ámbito léxico

Pila de Activación

Árbol de Activación

Ejecución con Ámbito léxico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

      Llamada a f
      Llamada a h
      Si condición = verdadero
         entonces Llamada a g
         si no    Uso de x
     fin si
    Uso de x
    Llamada a f
    Llamada a g
```

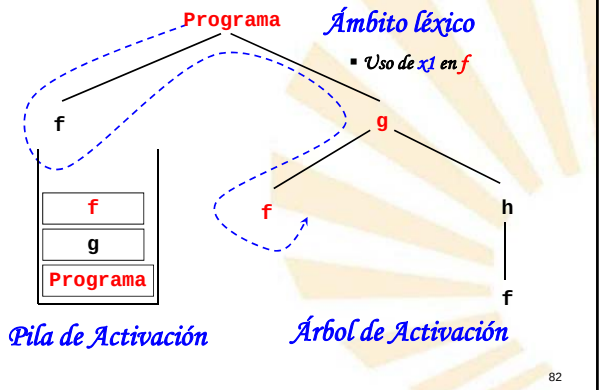


Ejecución con Ámbito léxico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)
     Declaración de procedimiento h
      Uso de x
      Llamada a f

     Llamada a f
     Llamada a h
     Si condición = verdadero
        entonces Llamada a g
        si no    Uso de x
     fin si
    Uso de x
    Llamada a f
    Llamada a g
```


Ejecución con Ámbito léxico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)
     Declaración de procedimiento h
      Uso de x
      Llamada a f

     Llamada a f
     Llamada a h
     Si condición = verdadero
        entonces Llamada a g
        si no    Uso de x
     fin si
    Uso de x
    Llamada a f
    Llamada a g
```

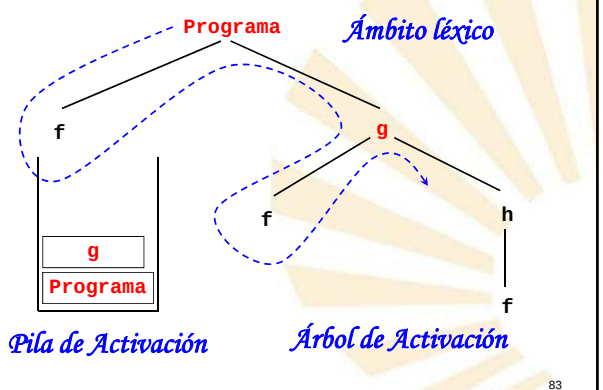


Ámbito léxico

- Uso de x1 en f

Pila de Activación

Árbol de Activación



Ámbito léxico

Pila de Activación

Árbol de Activación

Ejecución con Ámbito léxico

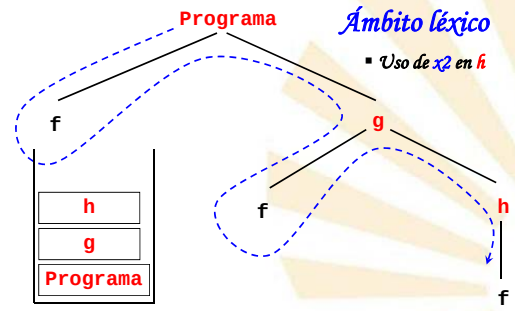
```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

       Llamada a f
       Llamada a h
       Si condición = verdadero
          entonces Llamada a g
          si no    Uso de x
       fin si
      Uso de x
      Llamada a f
      Llamada a g
```

Ejecución con Ámbito léxico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

      Llamada a f
      Llamada a h
      Si condición = verdadero
         entonces Llamada a g
         si no    Uso de x
      fin si
     Uso de x
     Llamada a f
     Llamada a g
```


Ámbito léxico

- *Uso de x2 en h*



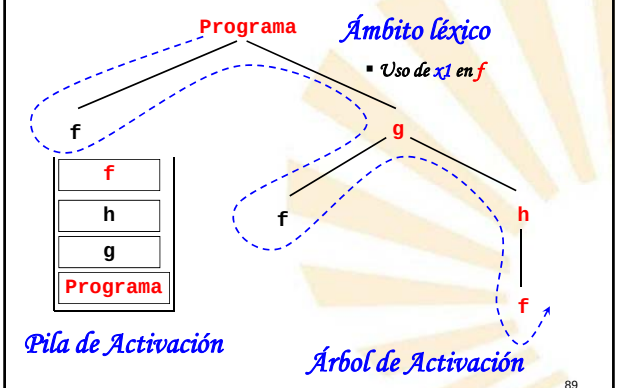
Pila de Activación *Árbol de Activación*

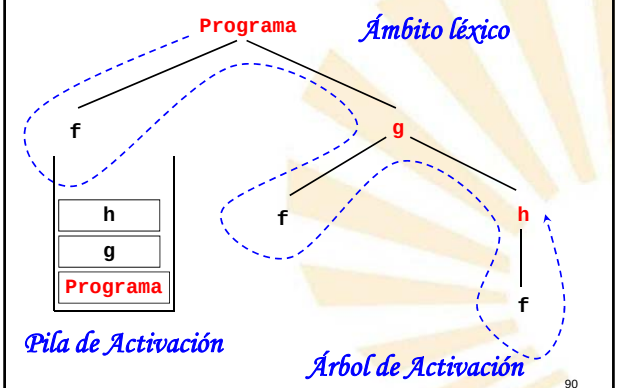
Ejecución con Ámbito léxico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

      Llamada a f
      Llamada a h
      Si condición = verdadero
         entonces Llamada a g
         si no    Uso de x
      fin si
     Uso de x
     Llamada a f
     Llamada a g
```

Ejecución con Ámbito léxico

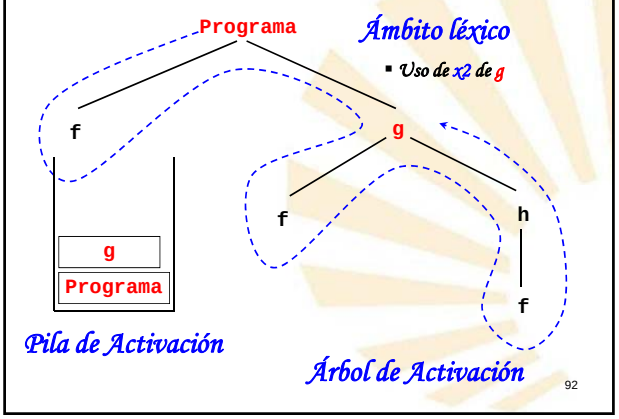
```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

       Llamada a f
       Llamada a h
       Si condición = verdadero
          entonces Llamada a g
          si no    Uso de x
       fin si
      Uso de x
      Llamada a f
      Llamada a g
```



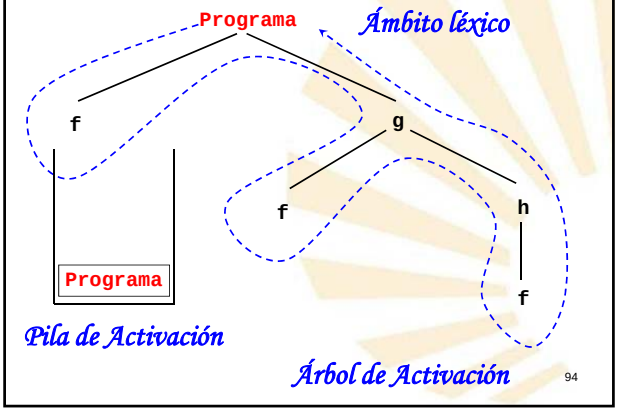
Ejecución con Ámbito léxico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

      Llamada a f
      Llamada a h
      Si condición = verdadero
         entonces Llamada a g
         si no    Uso de x
      fin si
     Uso de x
     Llamada a f
     Llamada a g
```

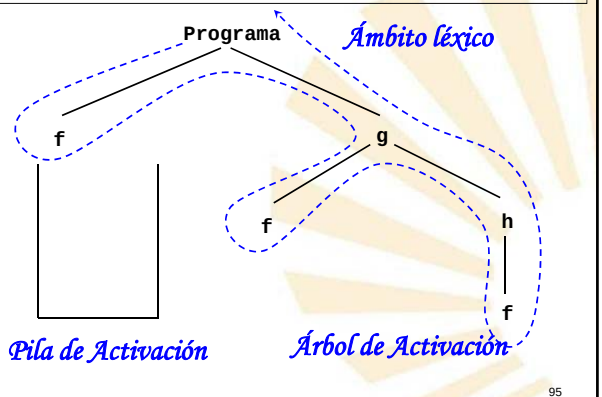



Ejecución con Ámbito léxico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

      Llamada a f
      Llamada a h
      Si condición = verdadero
         entonces Llamada a g
         si no    Uso de x
      fin si
     Uso de x
     Llamada a f
     Llamada a g
```



94

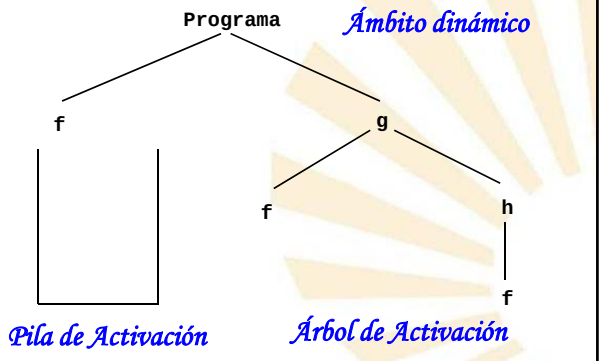


95

Ejecución con Ámbito dinámico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

      Llamada a f
      Llamada a h
      Si condición = verdadero
         entonces Llamada a g
         si no    Uso de x
      fin si
    Uso de x
    Llamada a f
    Llamada a g
```

96



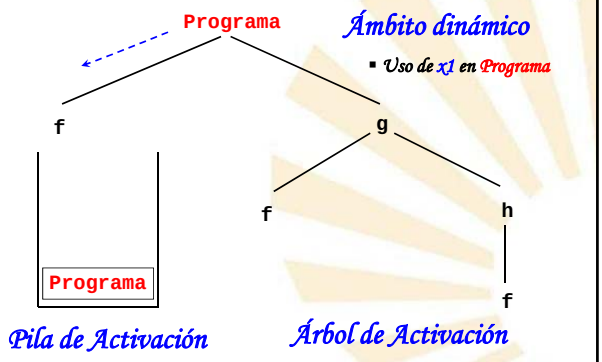
Ámbito dinámico

Pila de Activación

Árbol de Activación

Ejecución con Ámbito dinámico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

       Llamada a f
       Llamada a h
       Si condición = verdadero
          entonces Llamada a g
          si no    Uso de x
       fin si
      Uso de x
      Llamada a f
      Llamada a g
```

Ámbito dinámico

- Uso de *x1* en *Programa*

Pila de Activación

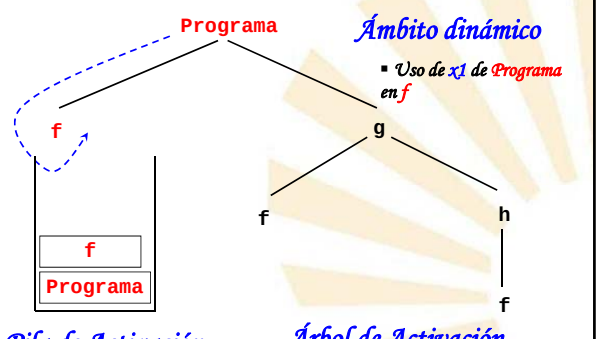
Árbol de Activación

Ejecución con Ámbito dinámico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

      Llamada a f
      Llamada a h
      Si condición = verdadero
         entonces Llamada a g
         si no    Uso de x
      fin si
     Uso de x
     Llamada a f
     Llamada a g
```


Ejecución con Ámbito dinámico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

      Llamada a f
      Llamada a h
      Si condición = verdadero
         entonces Llamada a g
         si no    Uso de x
      fin si
     Uso de x
     Llamada a f
     Llamada a g
```

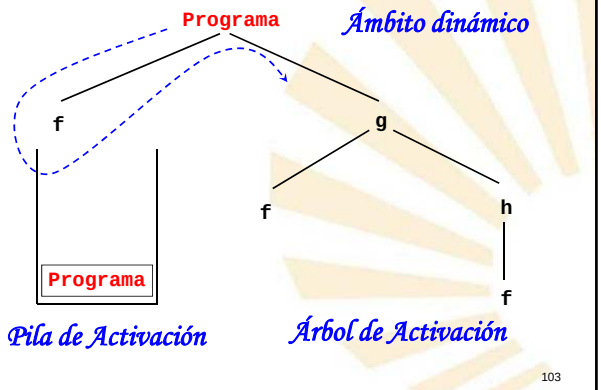



Ámbito dinámico

- *Uso de x1 de Programa en f*

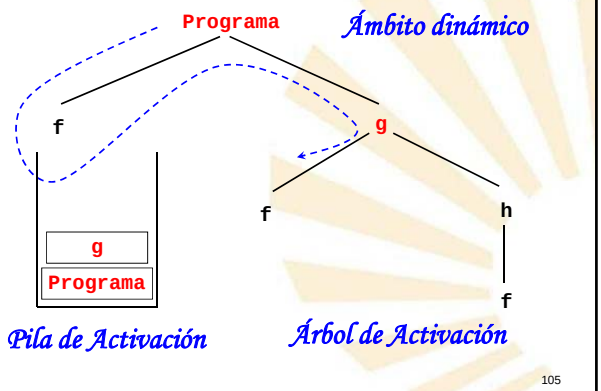
Pila de Activación

Árbol de Activación



Ejecución con Ámbito dinámico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

       Llamada a f
       Llamada a h
       Si condición = verdadero
          entonces Llamada a g
          si no    Uso de x
       fin si
      Uso de x
      Llamada a f
      Llamada a g
```

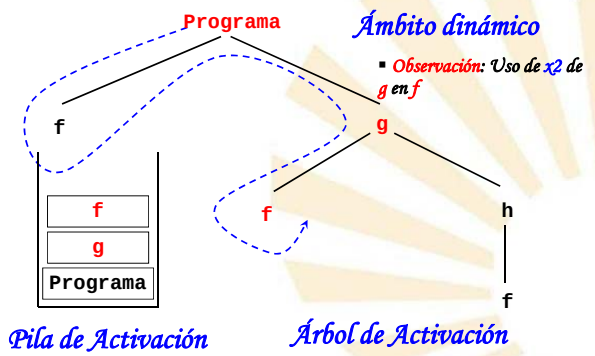



Ejecución con Ámbito dinámico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

      Llamada a f
      Llamada a h
      Si condición = verdadero
         entonces Llamada a g
         si no    Uso de x
      fin si
    Uso de x
    Llamada a f
    Llamada a g
```


Ejecución con Ámbito dinámico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

      Llamada a f
      Llamada a h
      Si condición = verdadero
         entonces Llamada a g
         si no    Uso de x
      fin si
    Uso de x
    Llamada a f
    Llamada a g
```

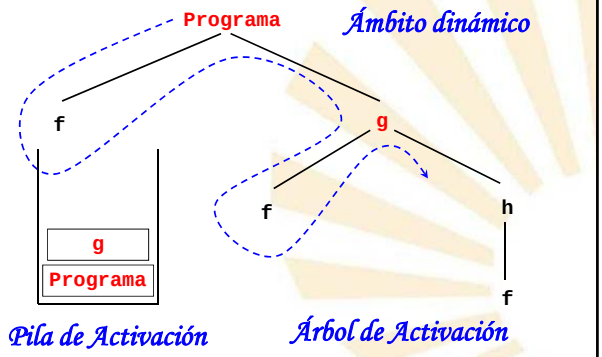



Ámbito dinámico

- *Observación: Uso de x2 de g en f*

Pila de Activación

Árbol de Activación

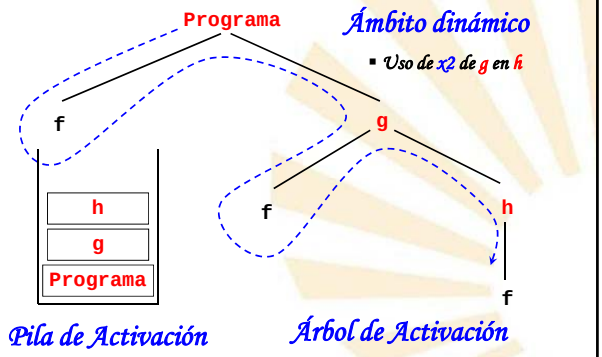


Ejecución con Ámbito dinámico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

       Llamada a f
       Llamada a h
       Si condición = verdadero
          entonces Llamada a g
          si no    Uso de x
       fin si
      Uso de x
      Llamada a f
      Llamada a g
```


Ejecución con Ámbito dinámico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

       Llamada a f
       Llamada a h
       Si condición = verdadero
          entonces Llamada a g
          si no    Uso de x
       fin si
      Uso de x
      Llamada a f
      Llamada a g
```



Ámbito dinámico

- Uso de x2 de g en h

Pila de Activación

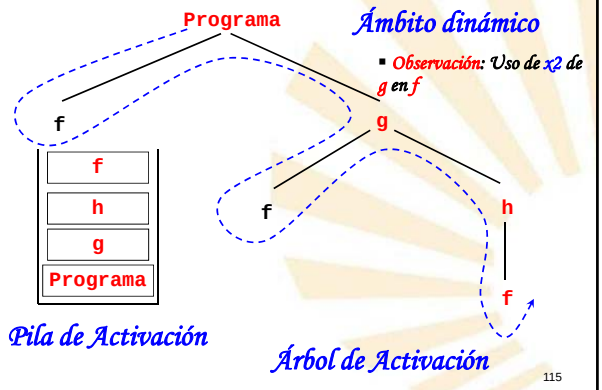
Árbol de Activación

Ejecución con Ámbito dinámico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

       Llamada a f
       Llamada a h
       Si condición = verdadero
          entonces Llamada a g
          si no    Uso de x
       fin si
      Uso de x
      Llamada a f
      Llamada a g
```


Ejecución con Ámbito dinámico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

       Llamada a f
       Llamada a h
       Si condición = verdadero
          entonces Llamada a g
          si no    Uso de x
       fin si
      Uso de x
      Llamada a f
      Llamada a g
```

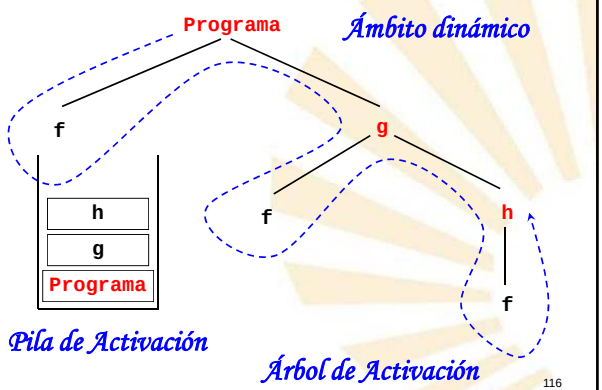


Ámbito dinámico

- Observación: Uso de x2 de g en f

Pila de Activación

Árbol de Activación



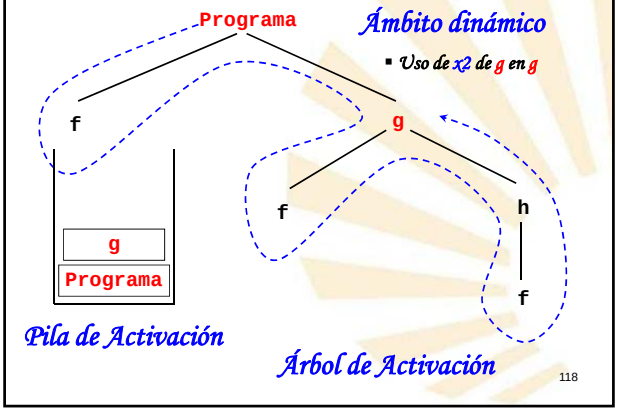
Ámbito dinámico

Pila de Activación

Árbol de Activación

Ejecución con Ámbito dinámico

```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

      Llamada a f
      Llamada a h
      Si condición = verdadero
         entonces Llamada a g
         si no    Uso de x
      fin si
    Uso de x
    Llamada a f
    Llamada a g
```



Ámbito dinámico

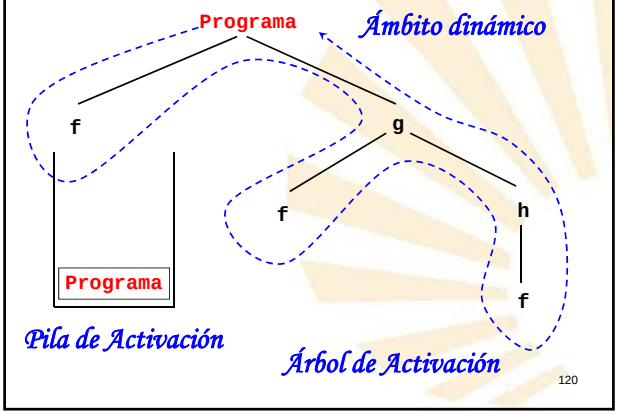
- Uso de x2 de g en g

Pila de Activación

Árbol de Activación

Ejecución con Ámbito dinámico

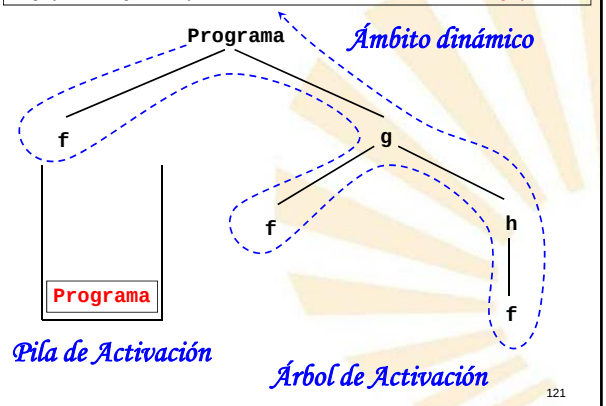
```
Programa
    Declaración de una variable x  (x1)

    Declaración de procedimiento f
     Uso de x

    Declaración de procedimiento g
     Declaración de una variable x  (x2)

     Declaración de procedimiento h
      Uso de x
      Llamada a f

       Llamada a f
       Llamada a h
       Si condición = verdadero
          entonces Llamada a g
          si no    Uso de x
       fin si
      Uso de x
      Llamada a f
      Llamada a g
```

Ámbito dinámico

Pila de Activación

Árbol de Activación



2. **Reseña Histórica de Scheme**
   - ✓ *LISP*
   - ✓ *Comparación entre Compilación e Interpretación*
   - ✓ *Comparación entre el ámbito léxico (o estático) y el dinámico*
   - ✓ **Origen de Scheme**

2. **Reseña Histórica de Scheme**
   - ✓ **Origen de Scheme**:
     - ➢ *Gerald Jay* **Sussman** *(MIT) and Guy Lewis* **Steele** *Jr.*
     - ➢ **Pregunta**:
       Cómo sería **LISP** *con reglas de Ámbito Léxico o Estático?*
     - ➢ **Respuesta**: *un nuevo lenguaje* → **Scheme**
       - *Implementación más* **eficiente** *de la* **recursión**
       - **Funciones** *de* **primera clase**
       - *Reglas* **semánticas** *rigurosas*
     - ➢ **Influencia en** *en Common LISP: reglas de Ámbito léxico*
     - ➢ **Revised^5 Report on the Algorithmic Language Scheme**

2. **Reseña Histórica de Scheme**
   - ✓ **Scheme**:
     - ➢ **Estructura de los programas de Scheme**
       - Secuencia de
         - **definiciones** de funciones y variables
         - y **expresiones**

UNIVERSIDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE
INFORMÁTICA Y ANÁLISIS NUMÉRICO

LENGUAJES
DE INTELIGENCIA ARTIFICIAL

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

SEGUNDO CURSO

PRIMER CUATRIMESTRE

CURSO ACADÉMICO 2009 - 2010