



UNIVERSIDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA

DEPARTAMENTO DE
INFORMÁTICA Y ANÁLISIS NUMÉRICO



PROGRAMACIÓN DECLARATIVA

INGENIERÍA INFORMÁTICA

CUARTO CURSO

PRIMER CUATRIMESTRE



Tema 1.- Introducción al Lenguaje Scheme

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 / 7)

- 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 / 7)**
 - **Características**
 - Expresividad
 - Extensible: regla del 10% - 90%
 - Protección
 - Elegancia Matemática

1. Características Fundamentales de la Programación Funcional

- Programación Declarativa (3 / 7)
 - Características
 - Expresividad
 - Muestra con claridad el significado de la programación
 - Extensible: regla del 10% - 90%
 - Protección
 - Elegancia Matemática

1. Características Fundamentales de la Programación Funcional

- Programación Declarativa (4 / 7)
 - Características
 - Expresividad
 - Extensible: regla del 10% - 90%
 - Facilidad para ampliar el lenguaje.
 - El 10 % del lenguaje permite resolver el 90% de los problemas.
 - Protección
 - Elegancia Matemática

1. Características Fundamentales de la Programación Funcional

- **Programación Declarativa (5 / 7)**
 - **Características**
 - Expresividad
 - Extensible: regla del 10% - 90%
 - **Protección**
 - ❑ El programador no tiene que preocuparse por detalles internos de gestión de memoria.
 - Elegancia Matemática

1. Características Fundamentales de la Programación Funcional

- Programación Declarativa (6 / 7)

- Características

- Expresividad
- Extensible: regla del 10% - 90%
- Protección
- Elegancia Matemática
 - Precisión y sencillez

1. Características Fundamentales de la Programación Funcional

- **Programación Declarativa (7 / 7)**
 - **Tipos:**
 - **Programación Funcional o Aplicativa:**
 - Lisp, **Scheme**, Haskell, Scala, Erlang,...
 - **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 expresiones o funciones más simples
 - **Ejecución de una función:**
 1. Evaluación de los **argumentos** o parámetros de la función
 2. Evaluación de las **expresiones** o **funciones** más simples
 3. Devolución del **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**

1. Características Fundamentales de la Programación Funcional

- **Gestión de la memoria**
 - **Gestión implícita de la memoria**
 - Tarea del intérprete.
 - El programador **no debe preocuparse** por la gestión de la memoria.
 - **Recolección de basura**
 - Tarea del intérprete.

1. Características Fundamentales de la Programación Funcional

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

Índice

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

2. Reseña Histórica de Scheme

- **LISP**

- **Contribuciones de LISP:**

- **Funciones “Built - in”**
- **Recolección de basura**
- **Lenguaje de Definición Formal:**
 - El propio lenguaje **LISP**

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.

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*): versión de la Costa Este

- 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.

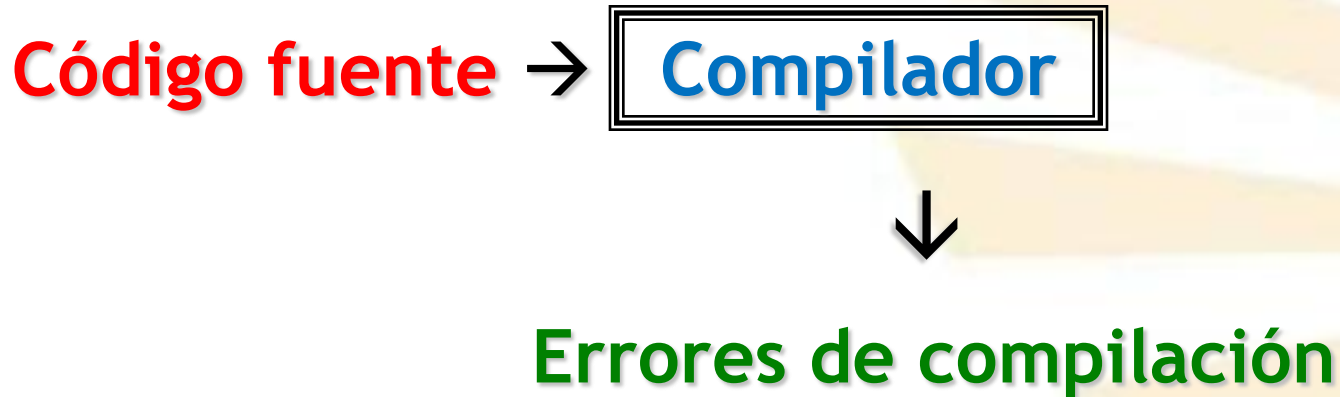
2. Reseña Histórica de Scheme

- Comparación entre Compilación e Interpretación
 - Compilación

Código fuente → **Compilador**

2. Reseña Histórica de Scheme

- Comparación entre Compilación e Interpretación
 - **Compilación**



2. Reseña Histórica de Scheme

- Comparación entre Compilación e Interpretación
 - Compilación

Código fuente → **Compilador** → **Código ejecutable**

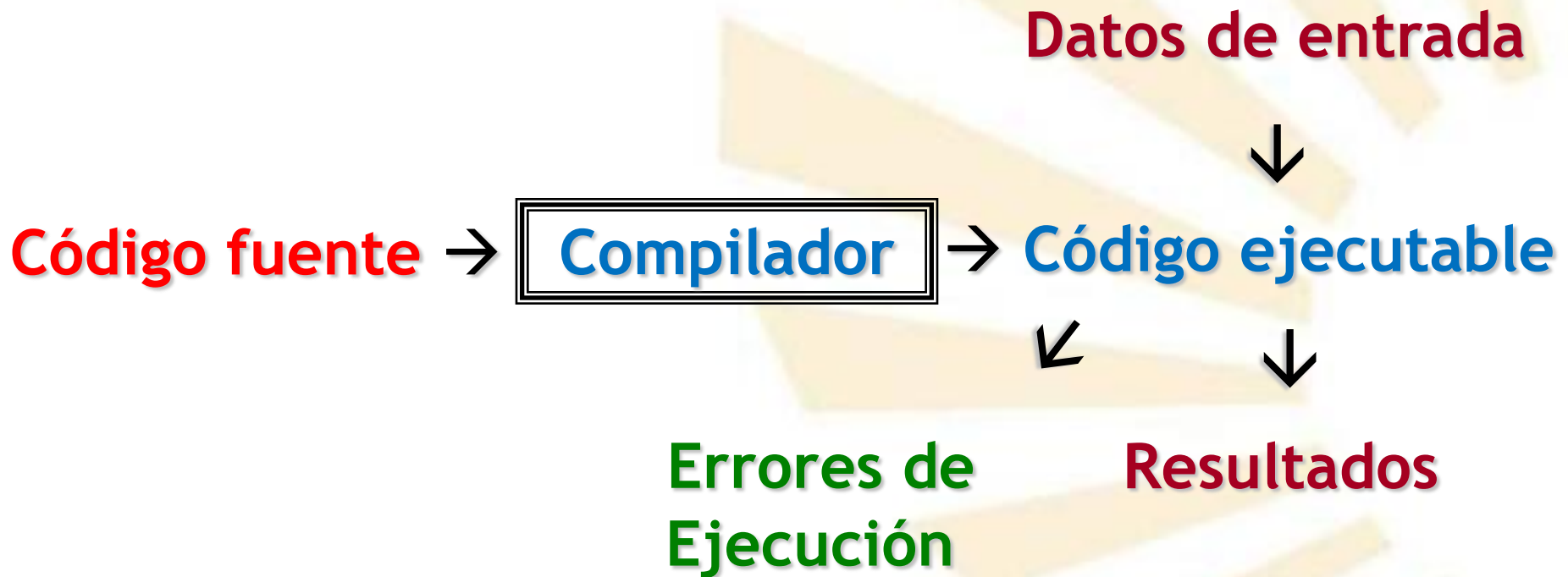
2. Reseña Histórica de Scheme

- Comparación entre Compilación e Interpretación
 - Compilación



2. Reseña Histórica de Scheme

- Comparación entre Compilación e Interpretación
 - Compilación



2. Reseña Histórica de Scheme

- Comparación entre Compilación e Interpretación
 - Compilación



2. Reseña Histórica de Scheme

- **Comparación entre Compilación e Interpretación**
 - **Interpretación**

2. Reseña Histórica de Scheme

- **Comparación entre Compilación e Interpretación**
 - **Interpretación o simulación:** consiste en un ciclo de tres fases

2. Reseña Histórica de Scheme

- **Comparación entre Compilación e Interpretación**

- **Interpretación o simulación:** consiste en un ciclo de tres fases

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 de tres fases

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 de tres fases

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

Código fuente →

Intérprete

2. Reseña Histórica de Scheme

- Comparación entre Compilación e Interpretación
 - Interpretación



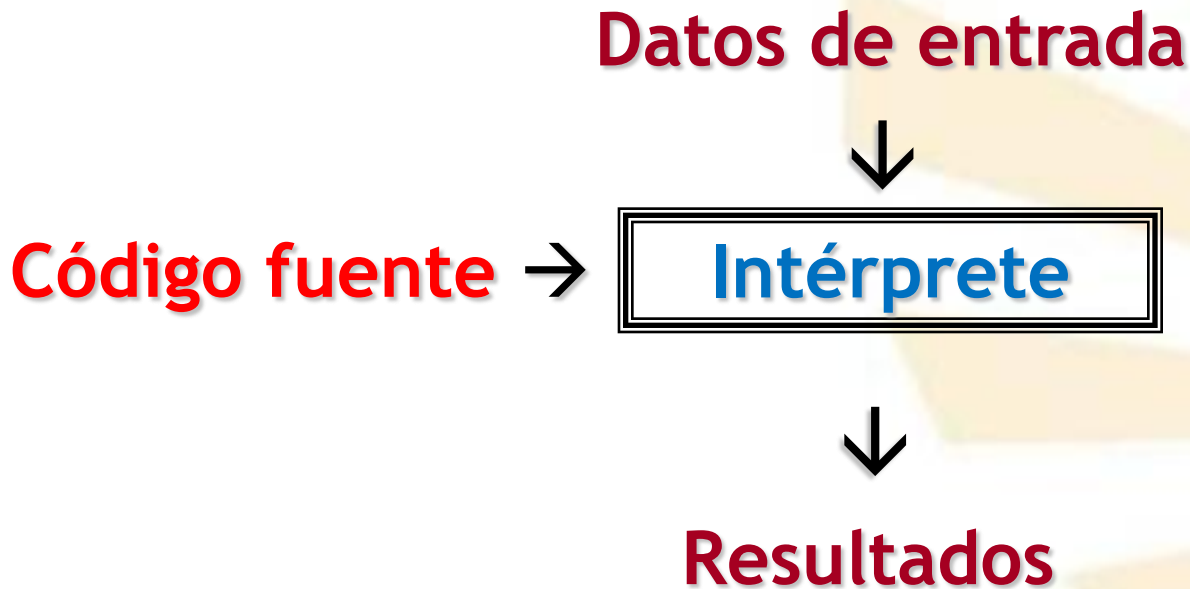
2. Reseña Histórica de Scheme

- Comparación entre Compilación e Interpretación
 - Interpretación



2. Reseña Histórica de Scheme

- Comparación entre Compilación e Interpretación
 - Interpretación



2. Reseña Histórica de Scheme

- Comparación entre Compilación e Interpretación

- **Compilación**

- Independencia
- Necesidades de memoria
- Eficiencia
- Global
- No interactividad
- Código **cerrado** durante la ejecución

- **Interpretación**

- Dependencia
- Sin necesidades de memoria
- Menos eficiencia
- Local
- Interactividad
- 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 **son usados** en otra función o procedimiento
 - donde **no han sido declarados.**

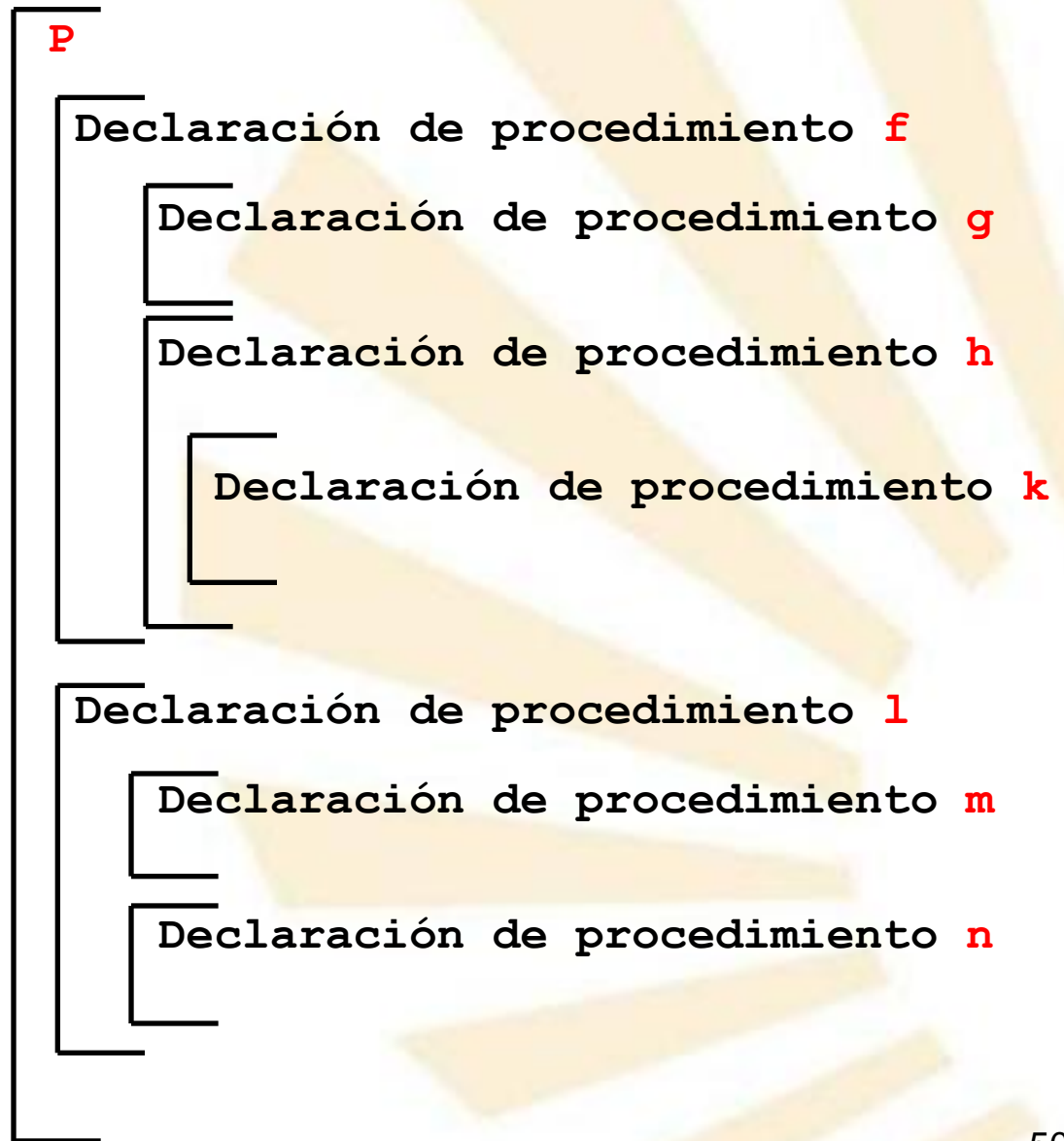
2. Reseña Histórica de Scheme

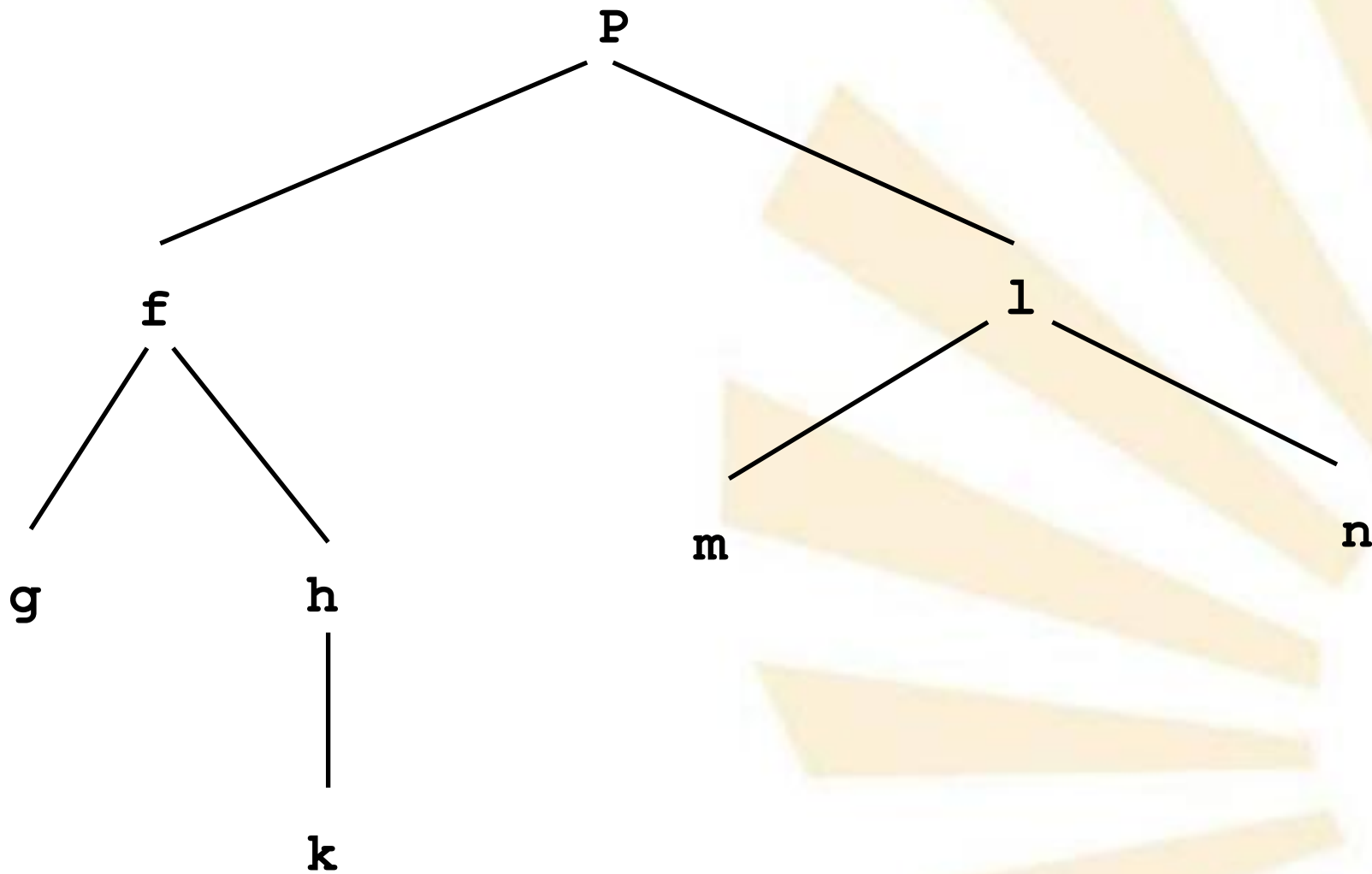
- **Comparación entre el ámbito léxico (o estático) y el dinámico**
 - **Tipos**
 - **Ámbito Léxico o Estático**
 - ❑ Con “estructura de bloques”: Pascal, **Scheme**
 - ❑ Sin “estructura de bloques”: C, Fortran
 - **Ámbito Dinámico**
 - ❑ Siempre con “estructura de bloques”:
 - Lisp, SNOBOL, APL

2. Reseña Histórica de Scheme

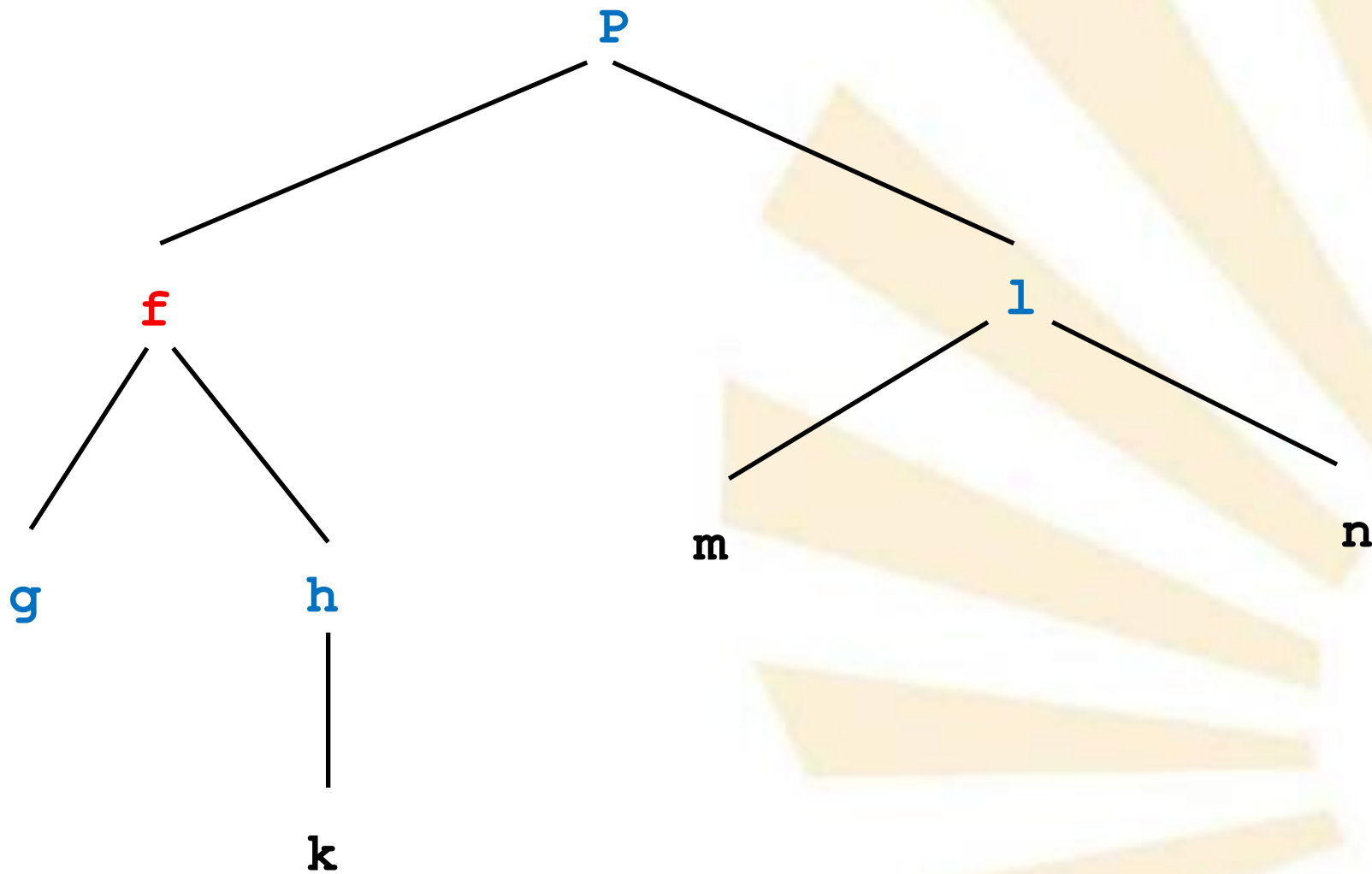
- Comparación entre el ámbito léxico (o estático) y el dinámico
 - Estructura de bloques
 - Un procedimiento o función puede **llamar** a
 - Sí mismo
 - Sus hijos (pero **no** a sus nietos...)
 - Sus hermanos (pero **no** a sus sobrinos)
 - Su padre, abuelo, bisabuelo, ...
 - Los hermanos de su padres,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

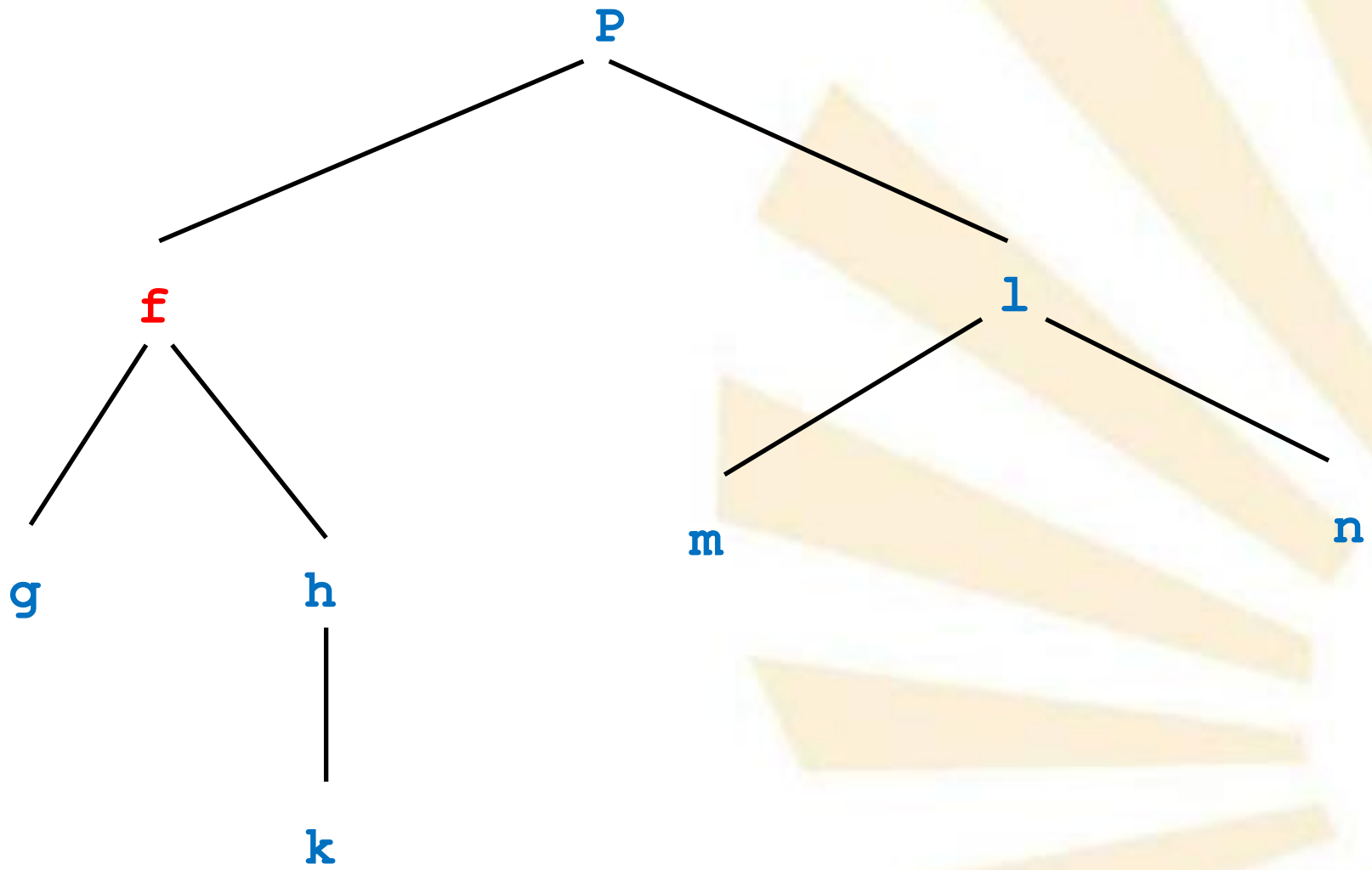




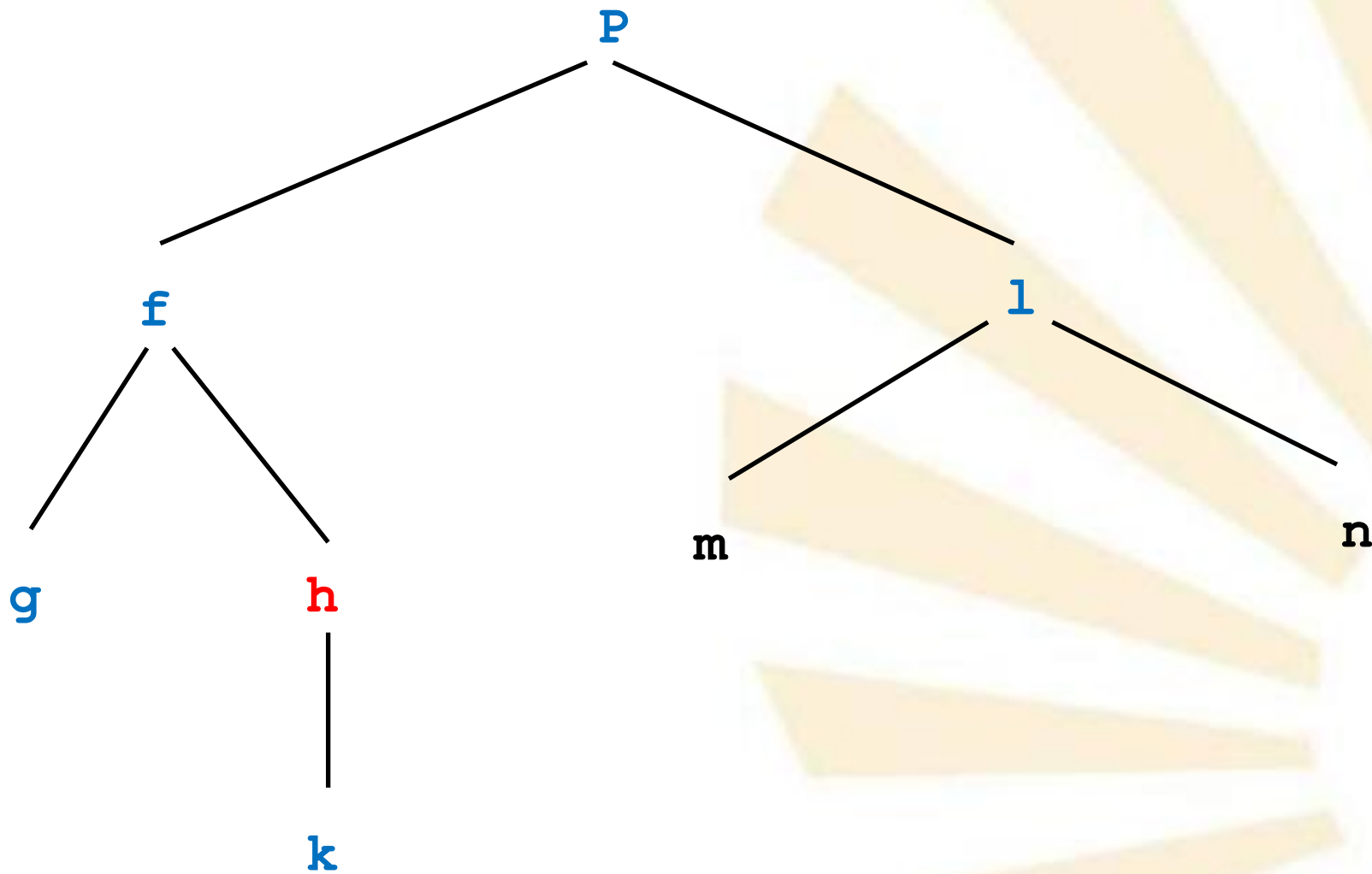
Jerarquía de la estructura de bloques



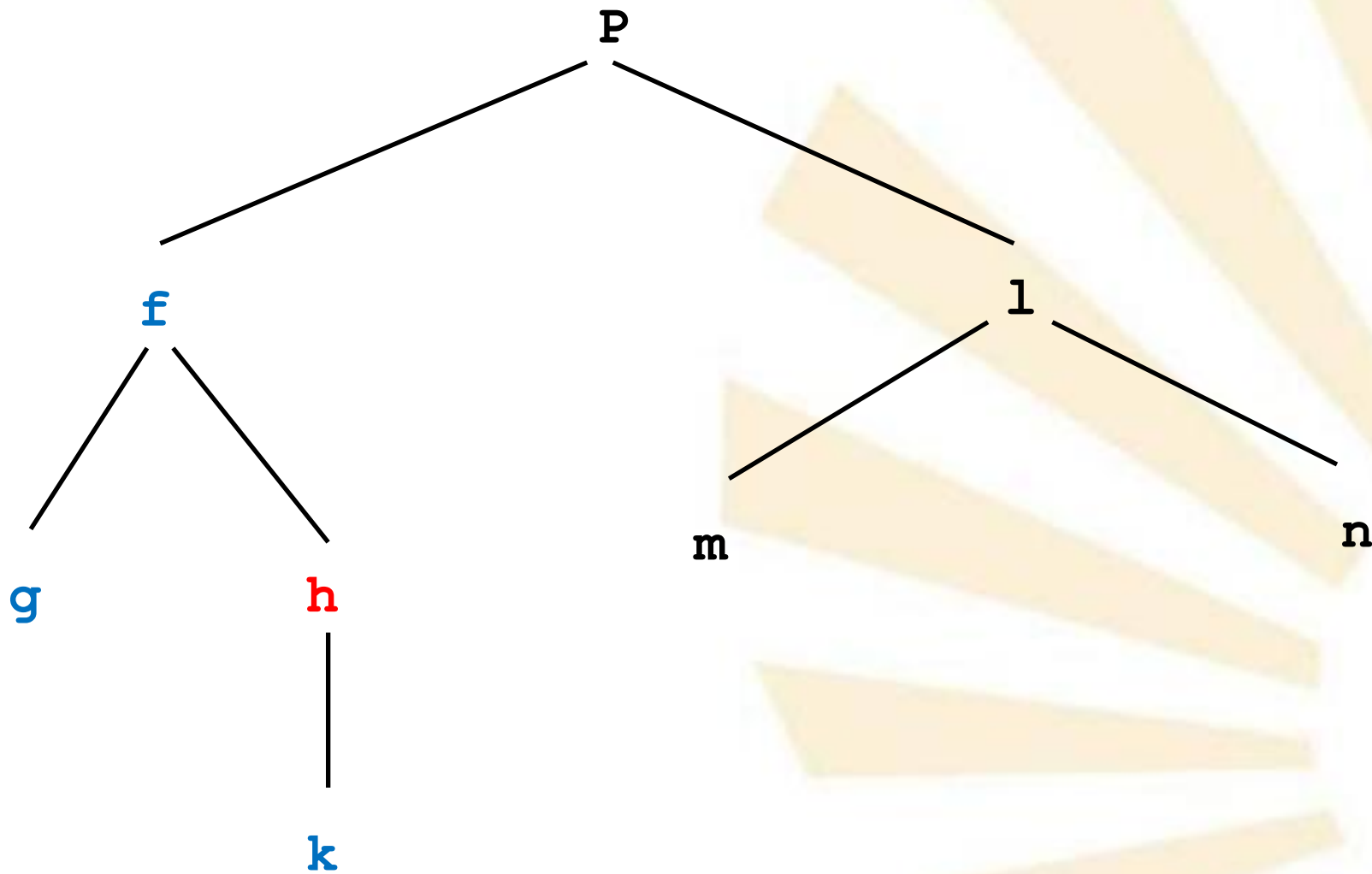
Funciones que pueden ser llamadas por **f**



Funciones que pueden llamar a **f**



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**
 - **Reglas del anidamiento más cercano**

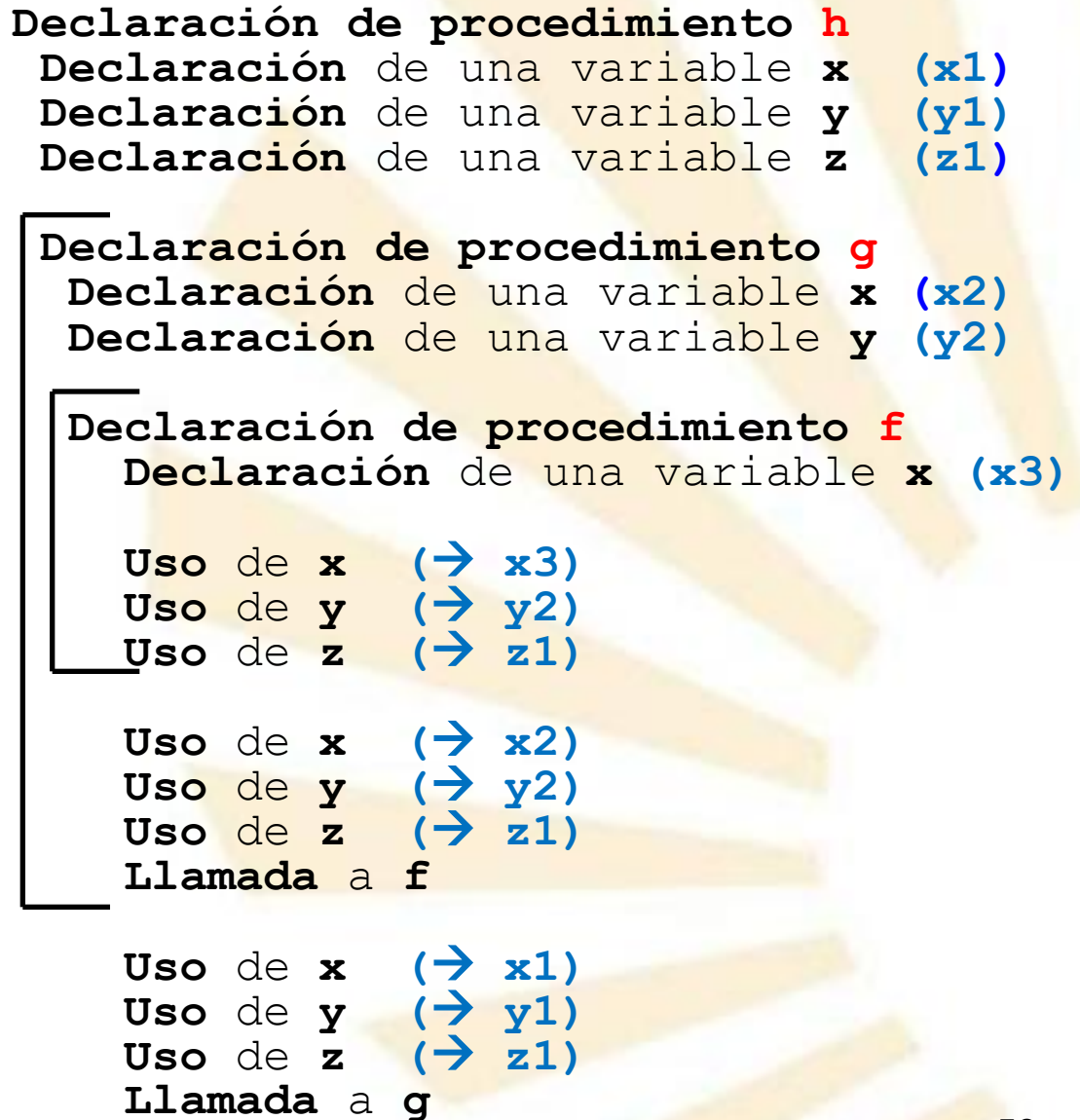
2. Reseña Histórica de Scheme

- **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 hay que **leer** el programa para determinar la **declaración** de un **identificador**

2. Reseña Histórica de Scheme

- Comparación entre el ámbito léxico (o estático) y el dinámico
 - **Ámbito léxico o estático**
 - **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”



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 **específica** 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 */
```

Las variables
globales no son
recomendables

```
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 */  
}
```

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.
 - Regla de **activación más cercana**.

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.
 - Hay que **ejecutar** el programa para determinar la declaración de un identificador

2. Reseña Histórica de Scheme

- Comparación entre el ámbito léxico (o estático) y el dinámico
 - **Ámbito dinámico**
 - 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

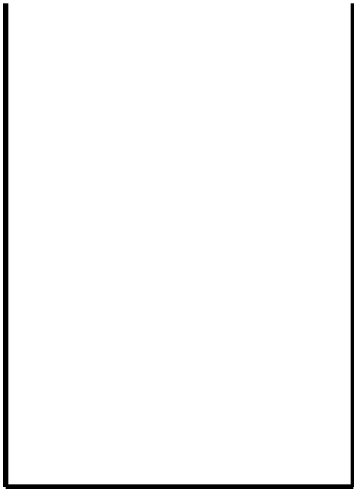
Uso de **x**

Llamada a **f**

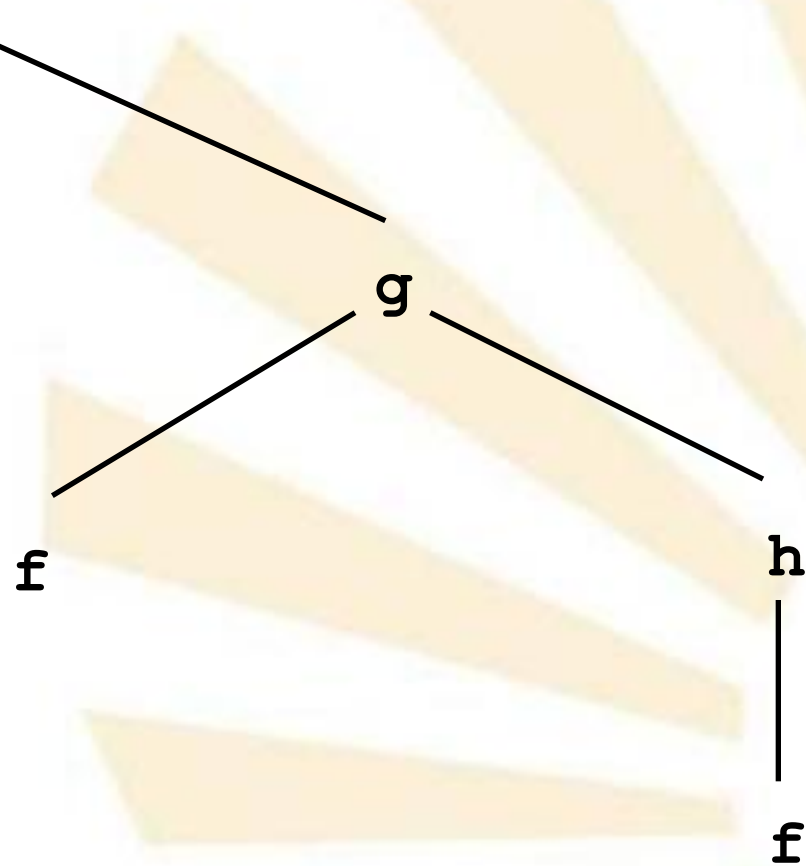
Llamada a **g**

Programa

f

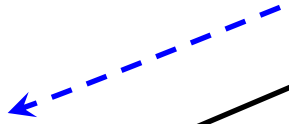


Pila de Activación



Árbol de Activación

Programa



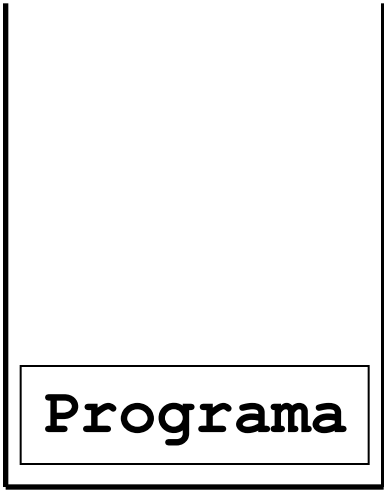
f

g

f

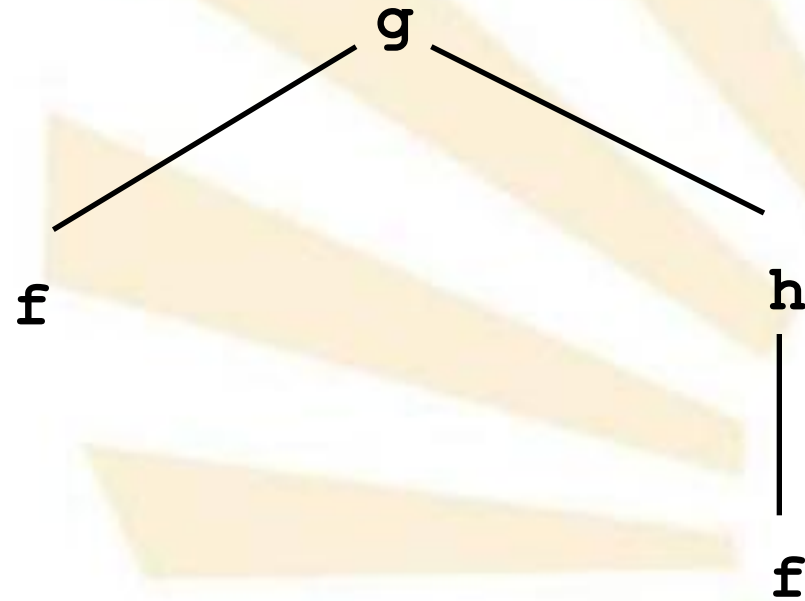
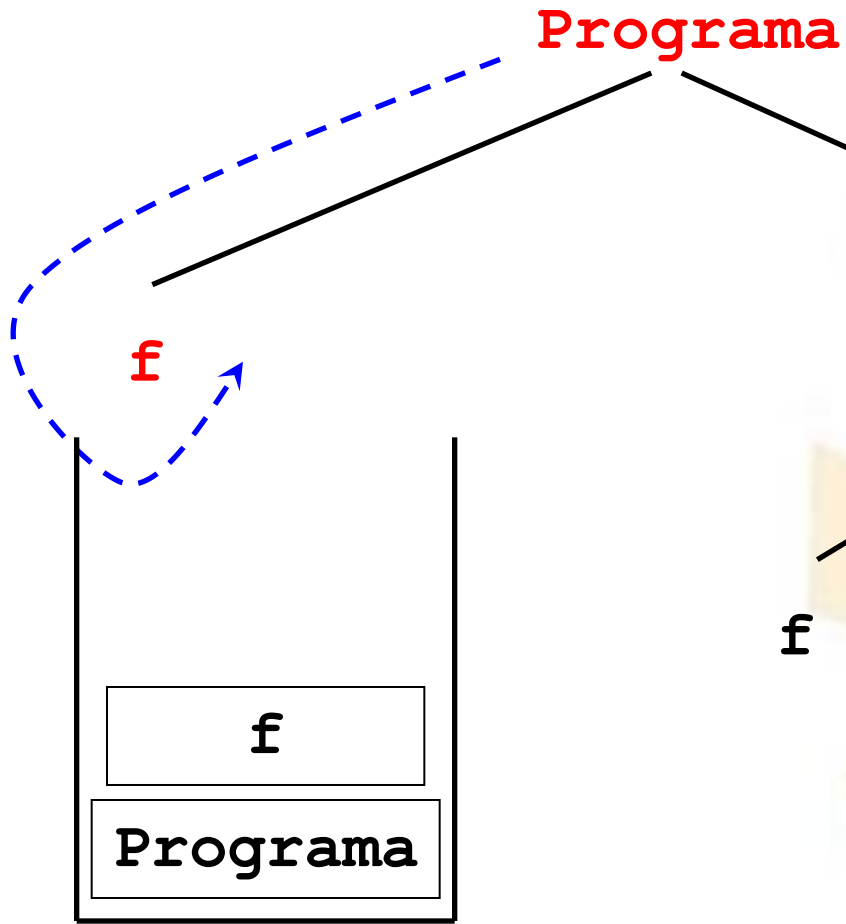
h

f



Pila de Activación

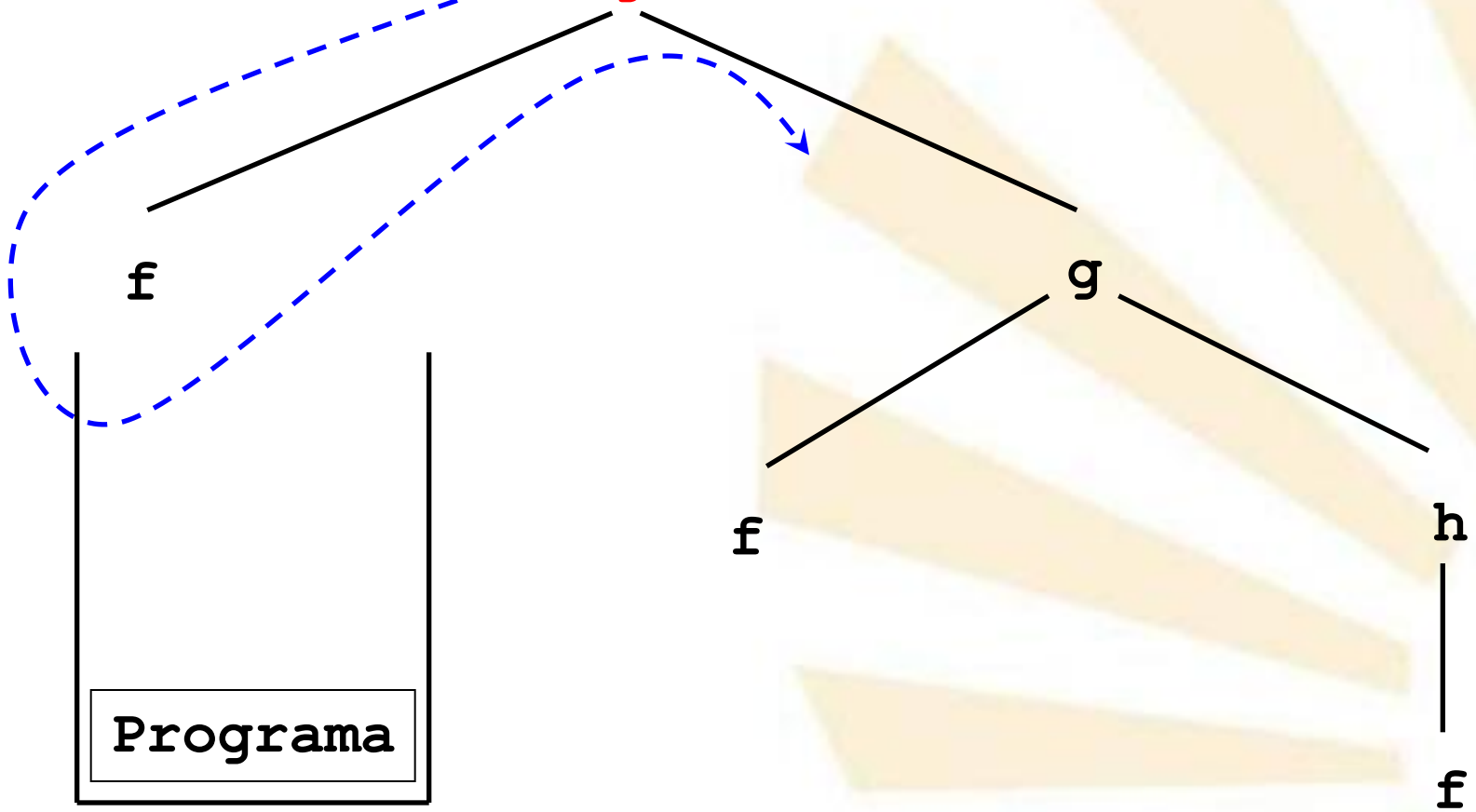
Árbol de Activación



Pila de Activación

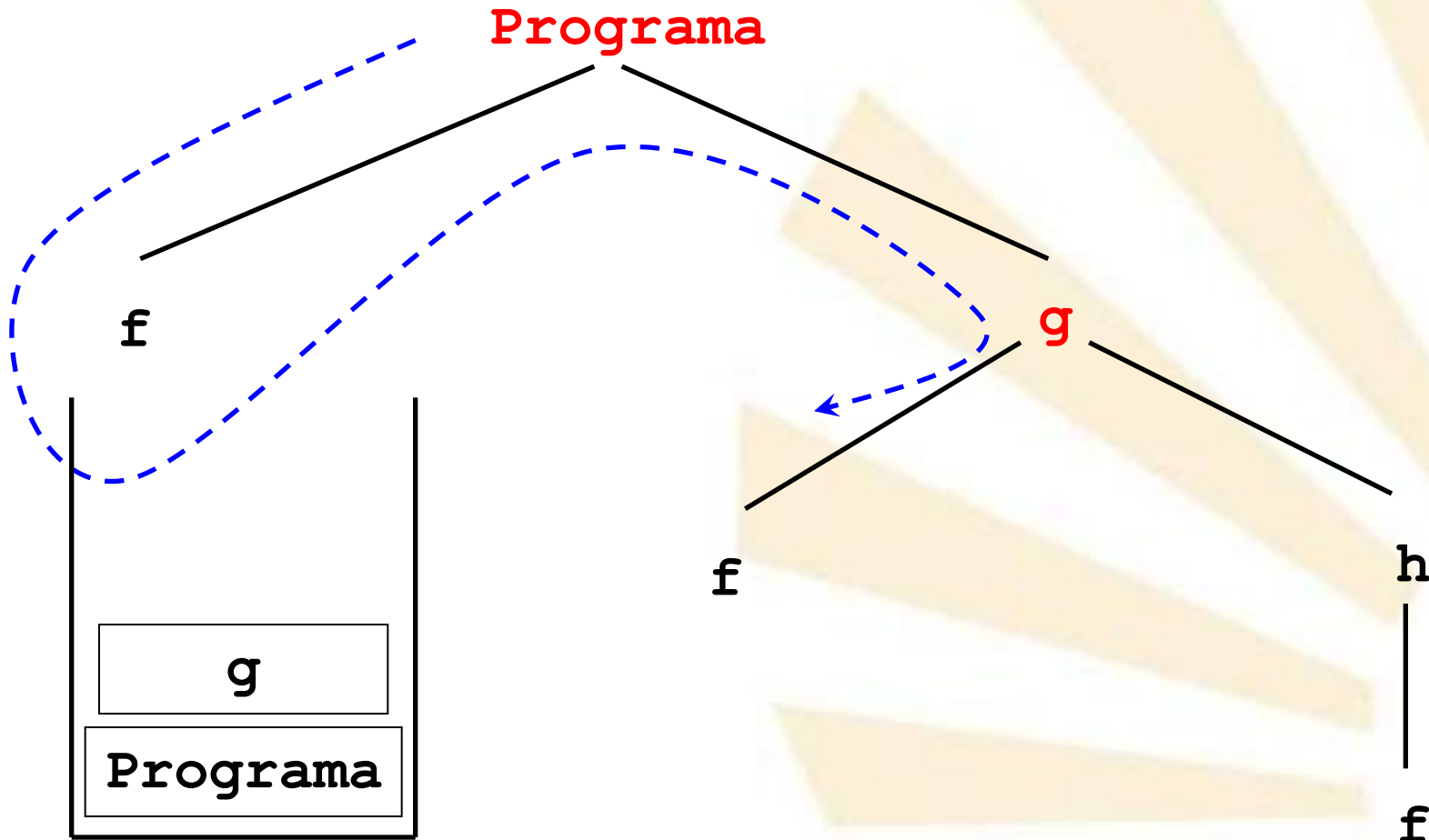
Árbol de Activación

Programa



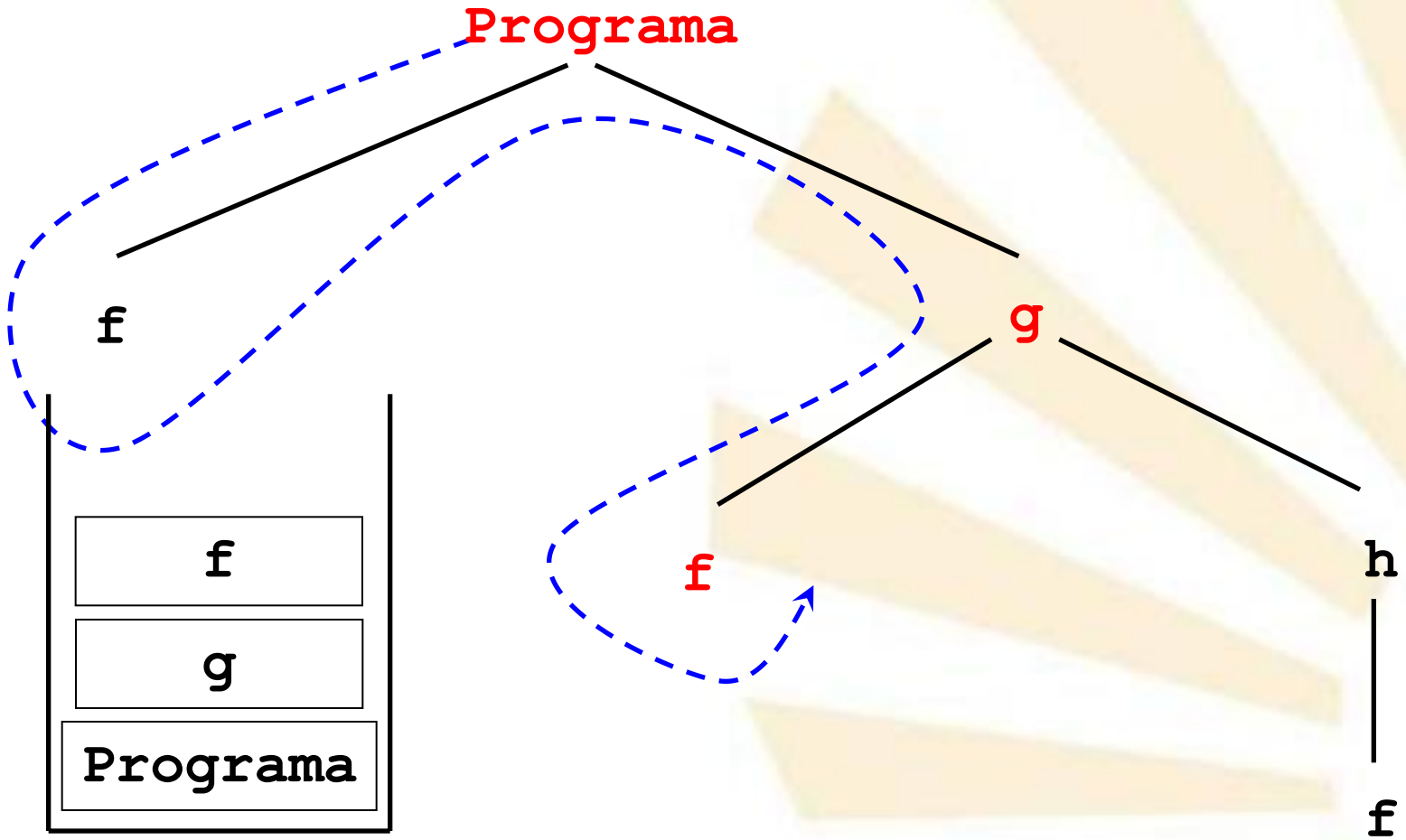
Pila de Activación

Árbol de Activación



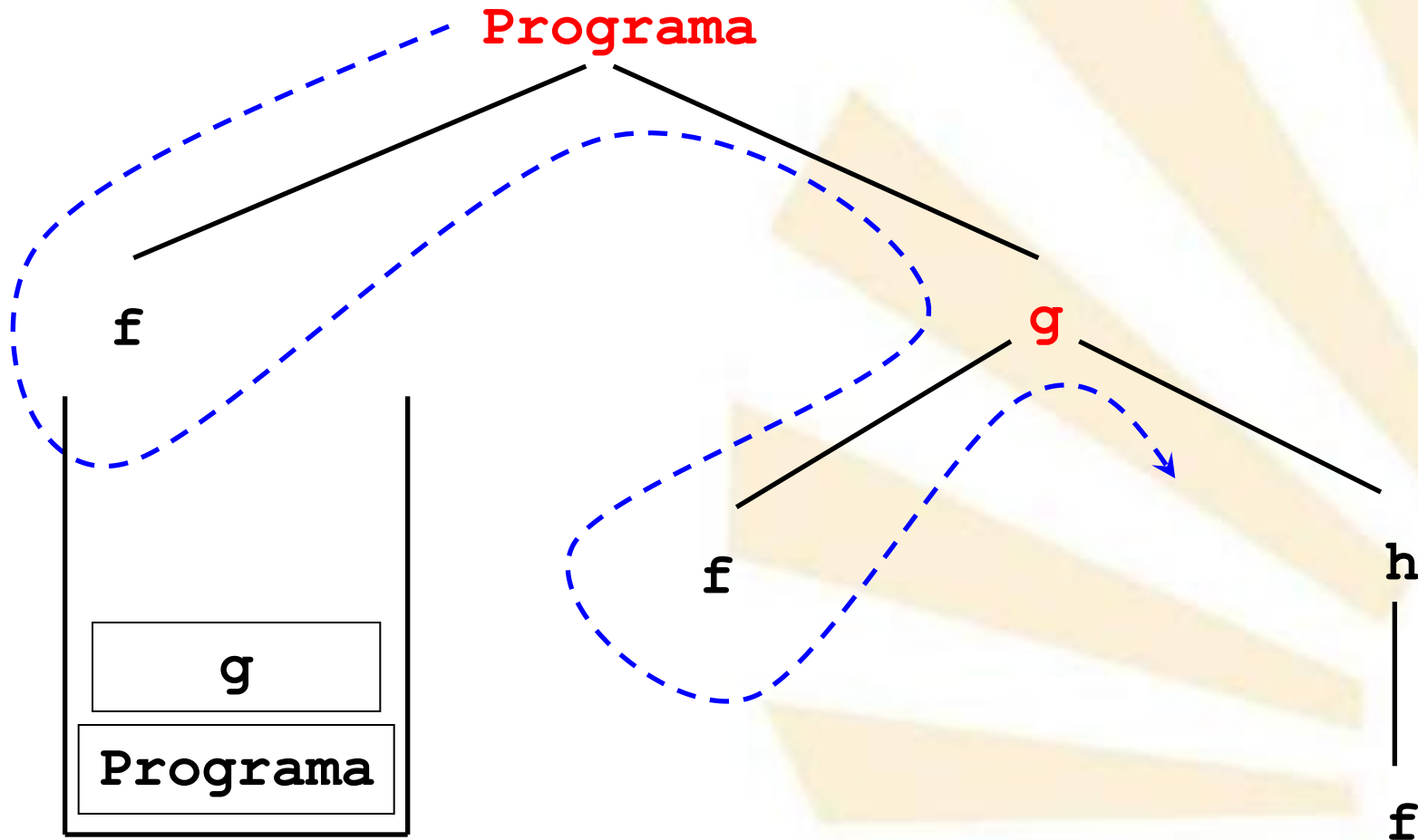
Pila de Activación

Árbol de Activación



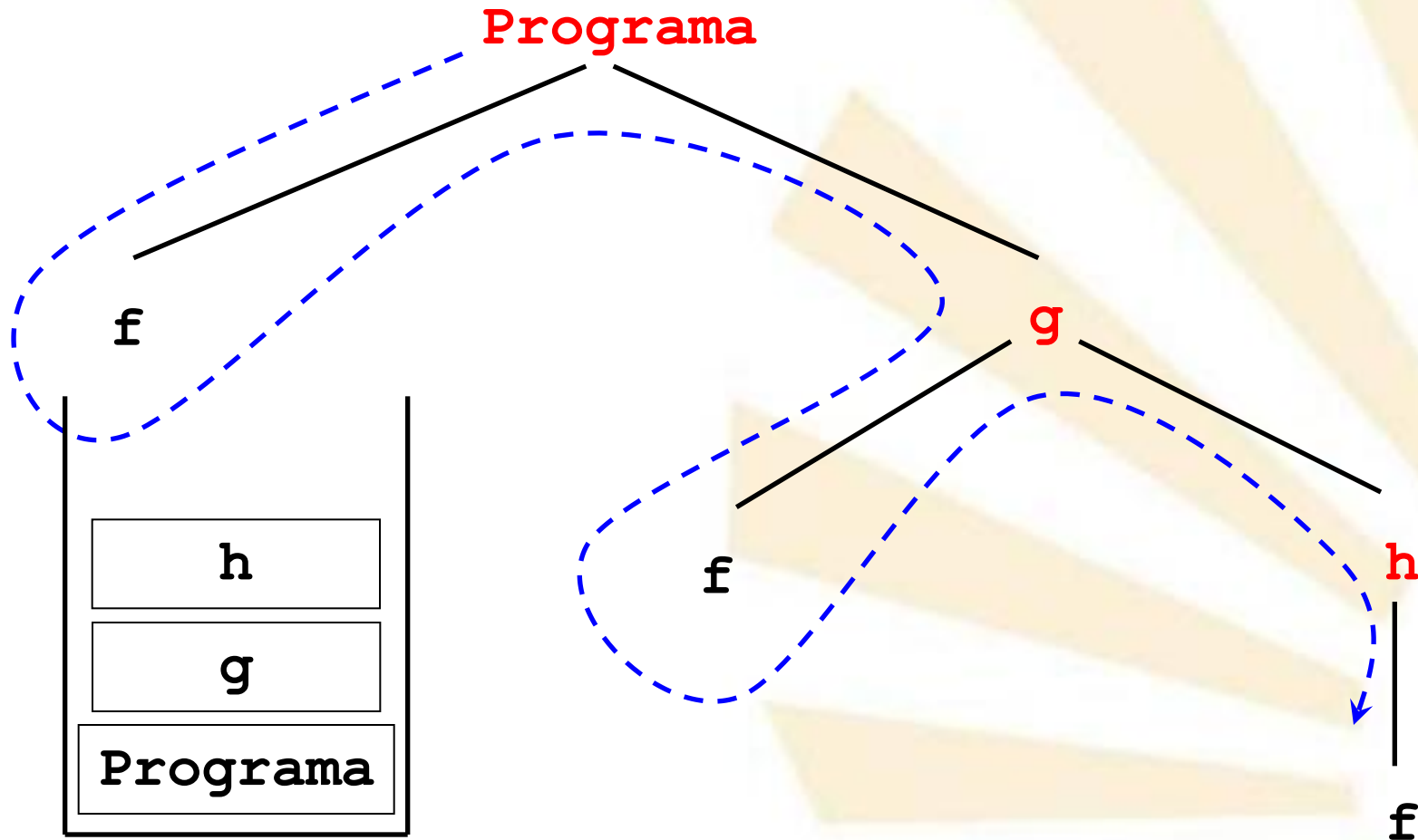
Pila de Activación

Árbol de Activación



Pila de Activación

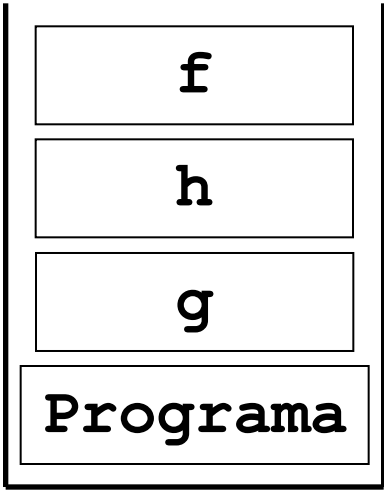
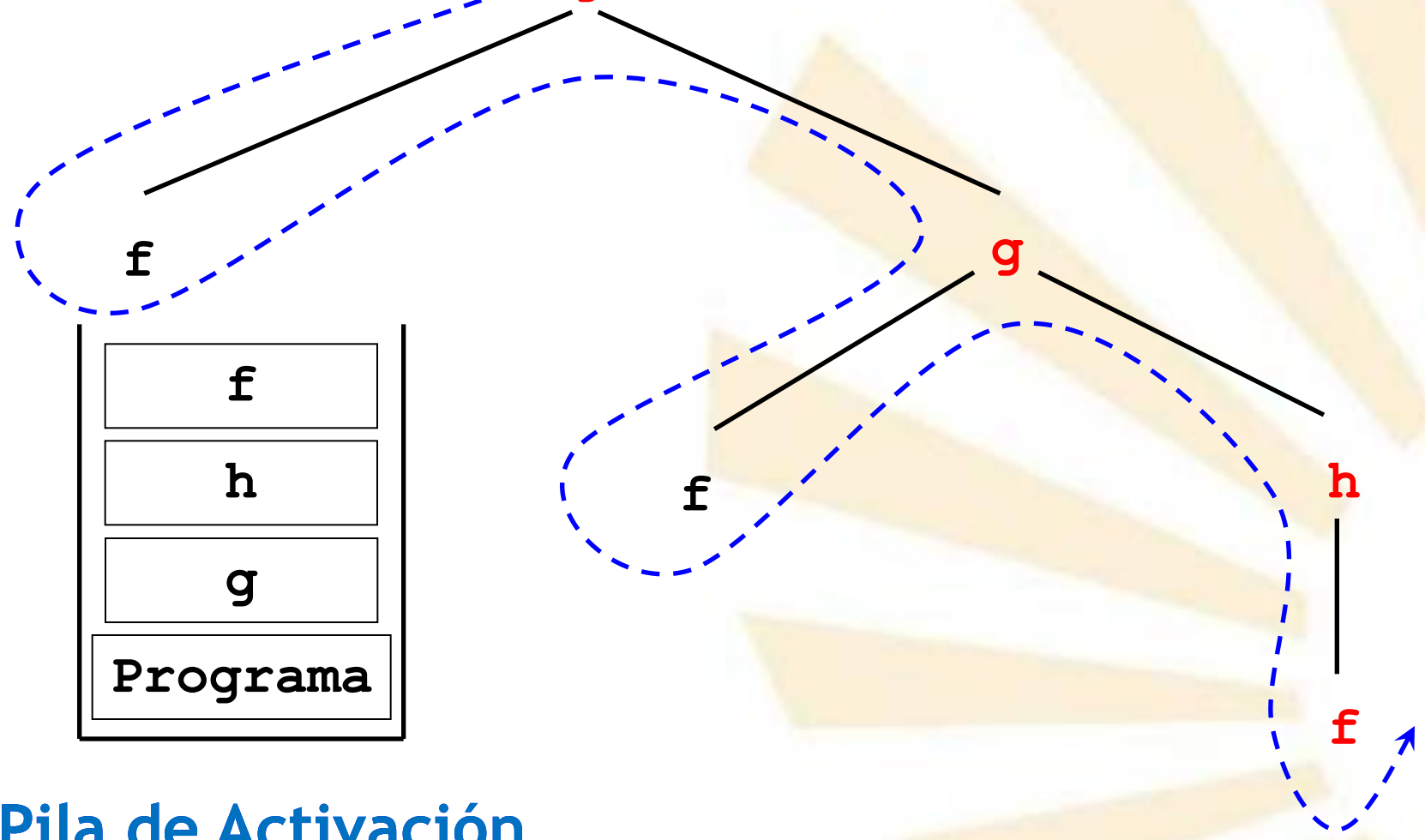
Árbol de Activación



Pila de Activación

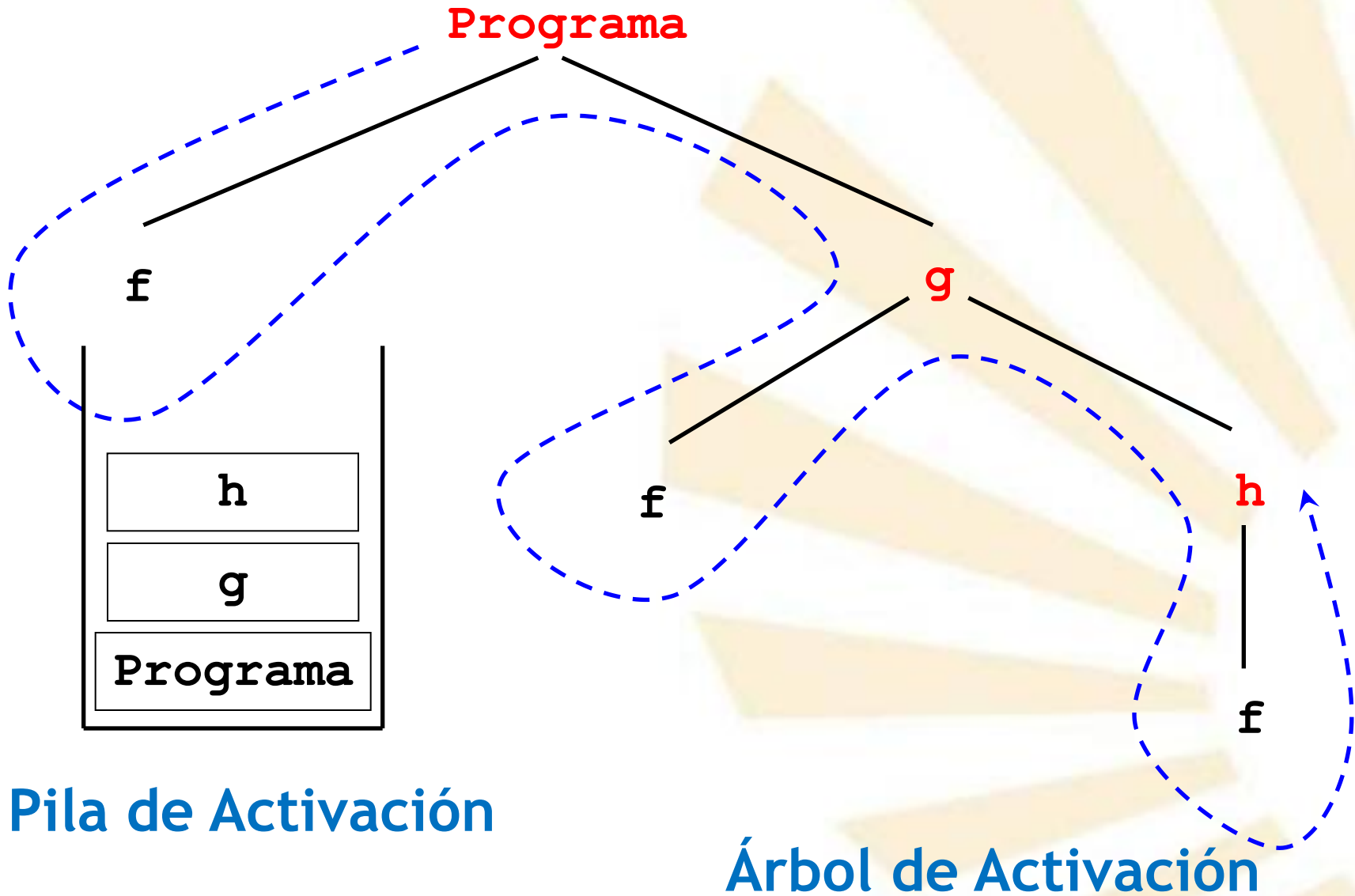
Árbol de Activación

Programa



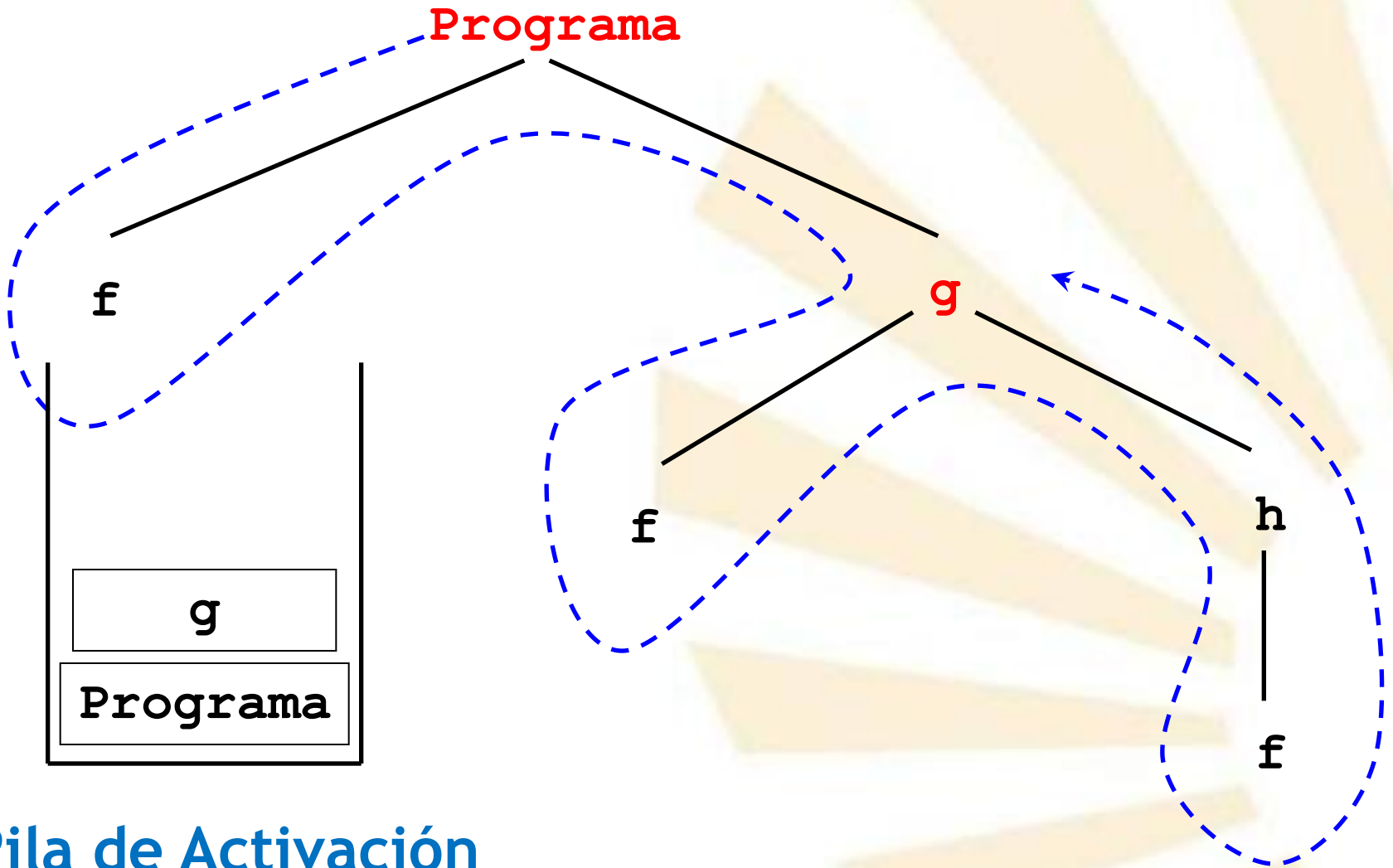
Pila de Activación

Árbol de Activación



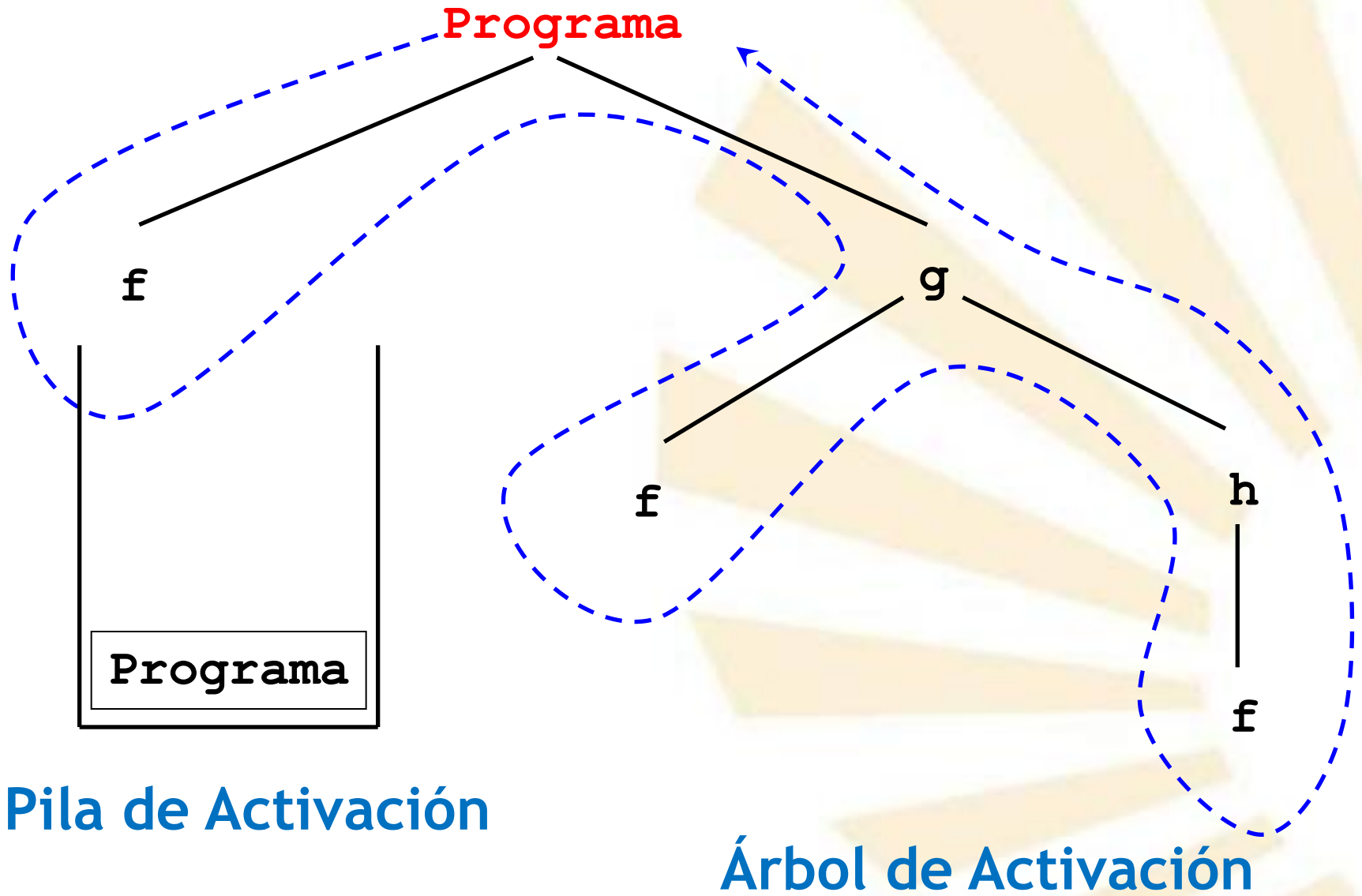
Pila de Activación

Árbol de Activación

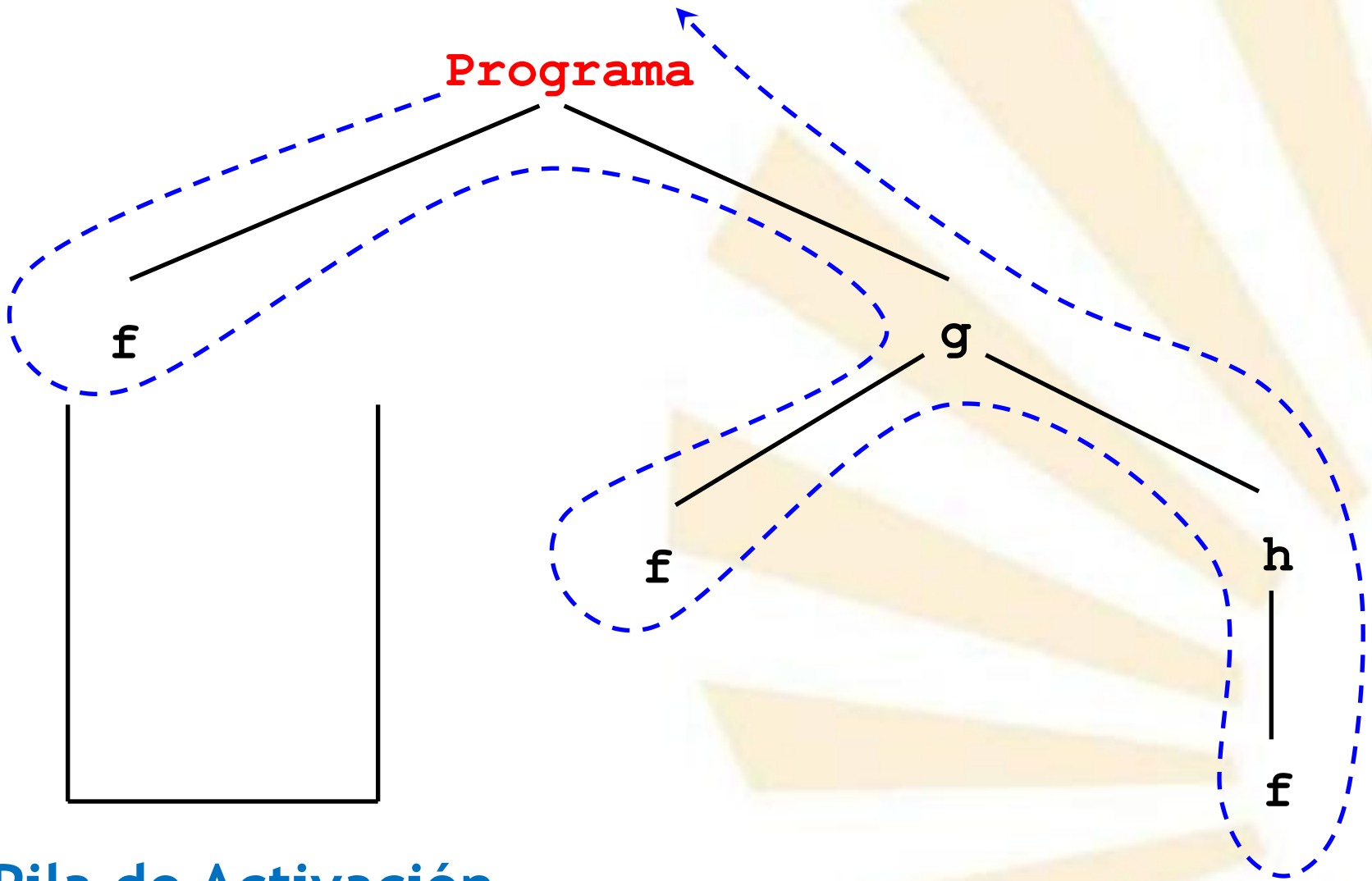


Pila de Activación

Árbol de Activación

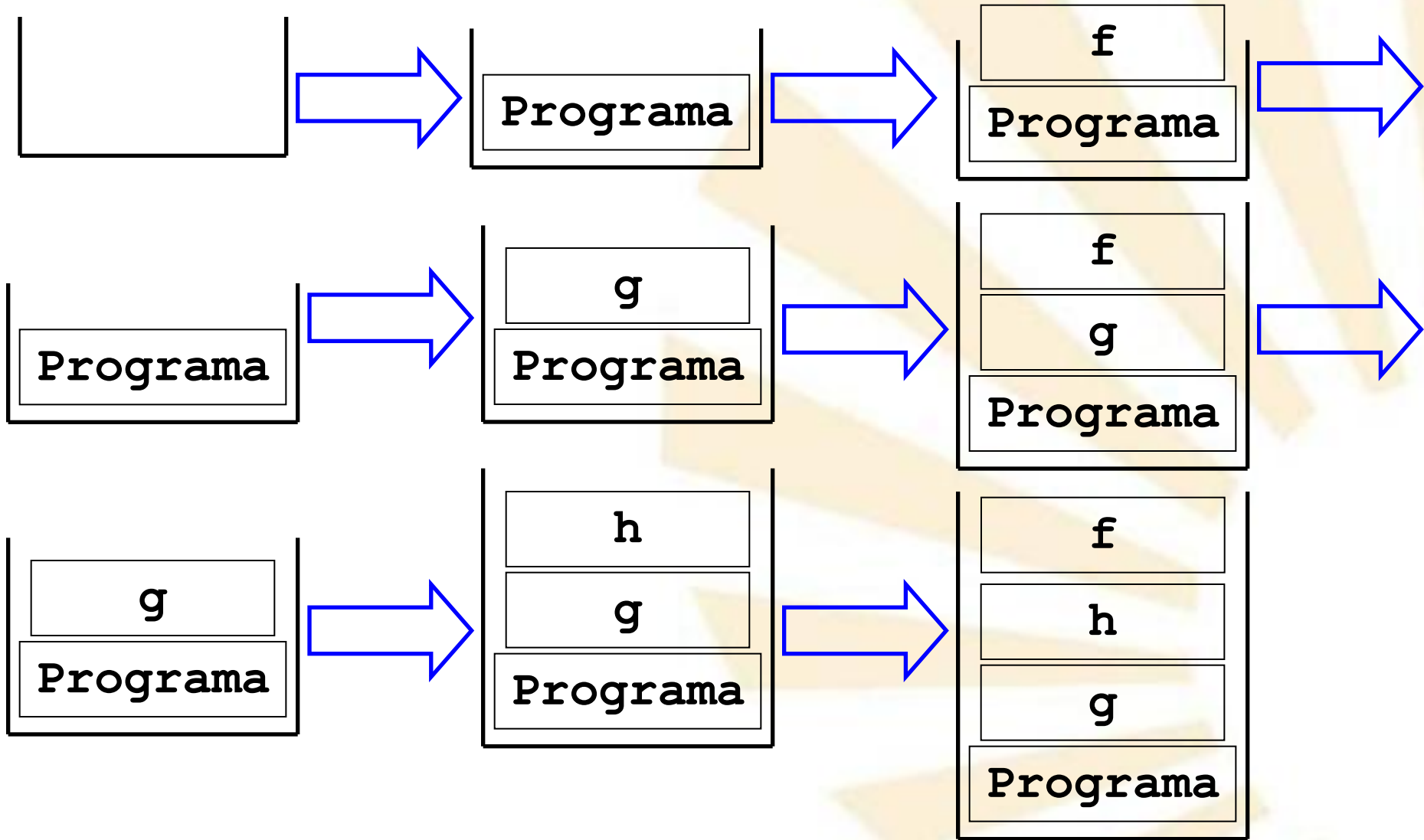


Programa

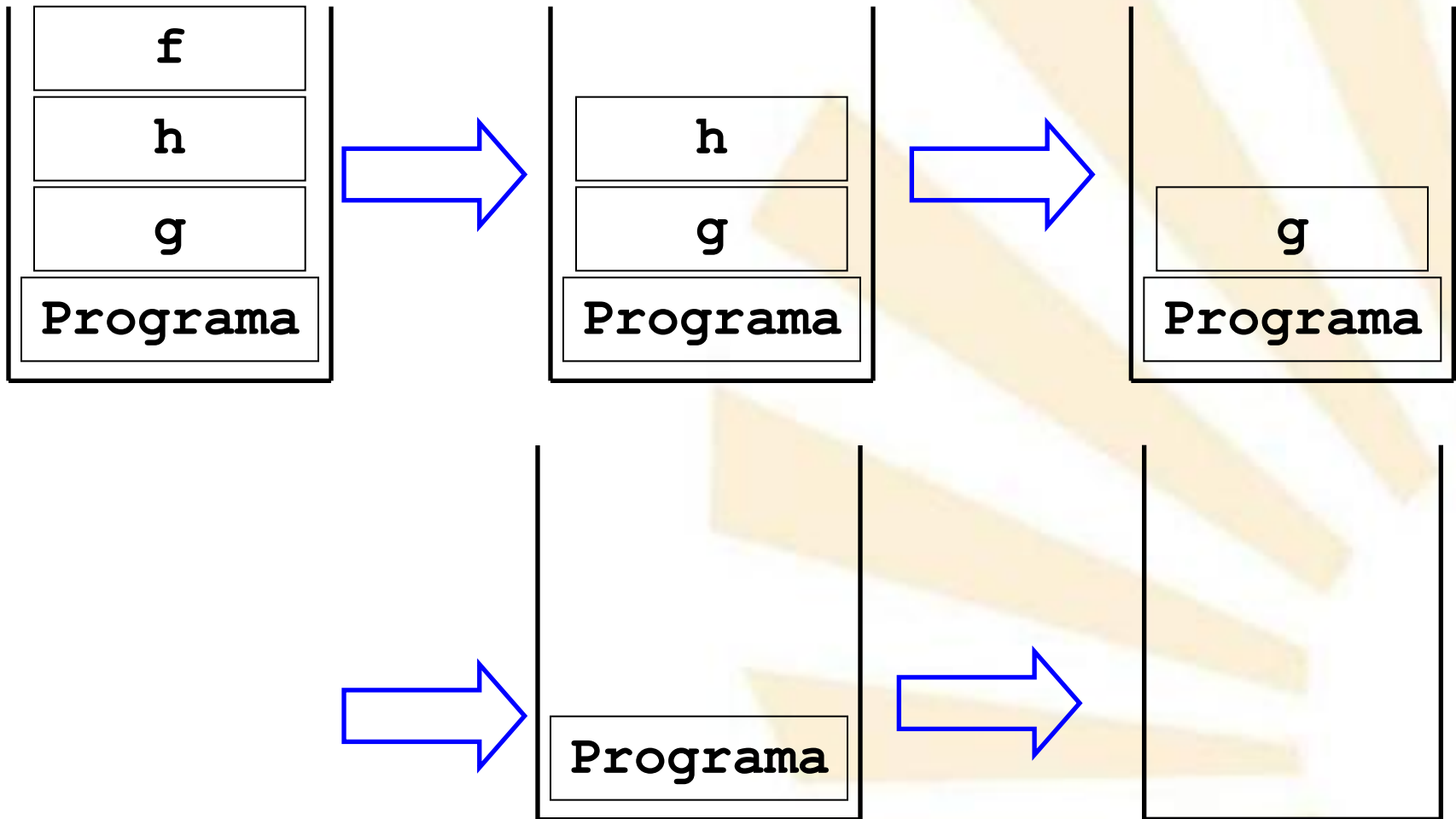


Pila de Activación

Árbol de Activación



Cambios en la Pila de Activación (1 / 2)



Cambios en la Pila de Activación (2 / 2)

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**

Programa

Ámbito léxico

f

g

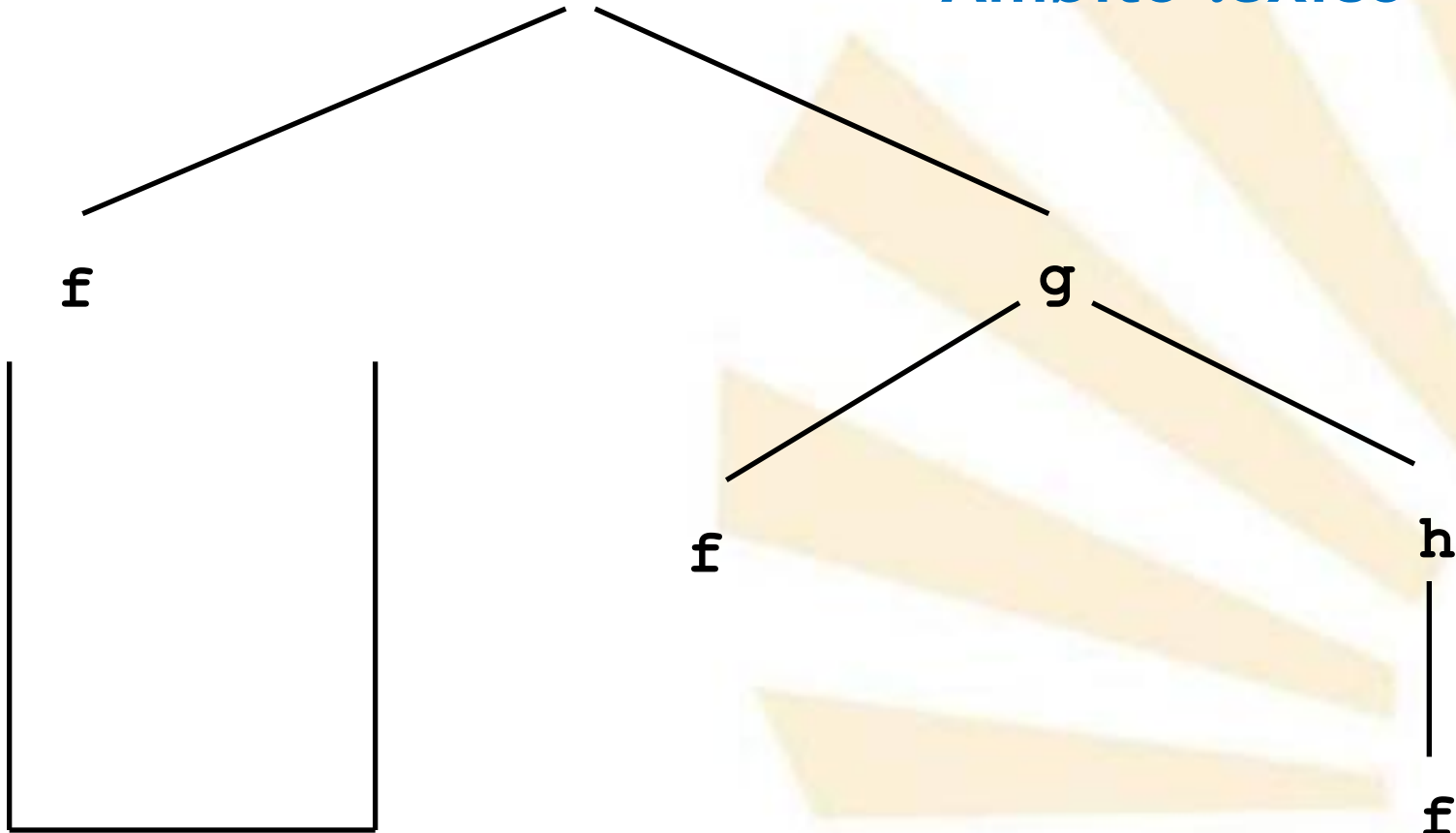
f

h

f

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**

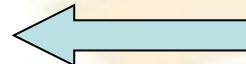
si no Uso de **x**

fin si

Uso de **x**

Llamada a **f**

Llamada a **g**



Programa

Ámbito léxico

- *Uso de x1 en Programa*

f

g

f

h

f

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**

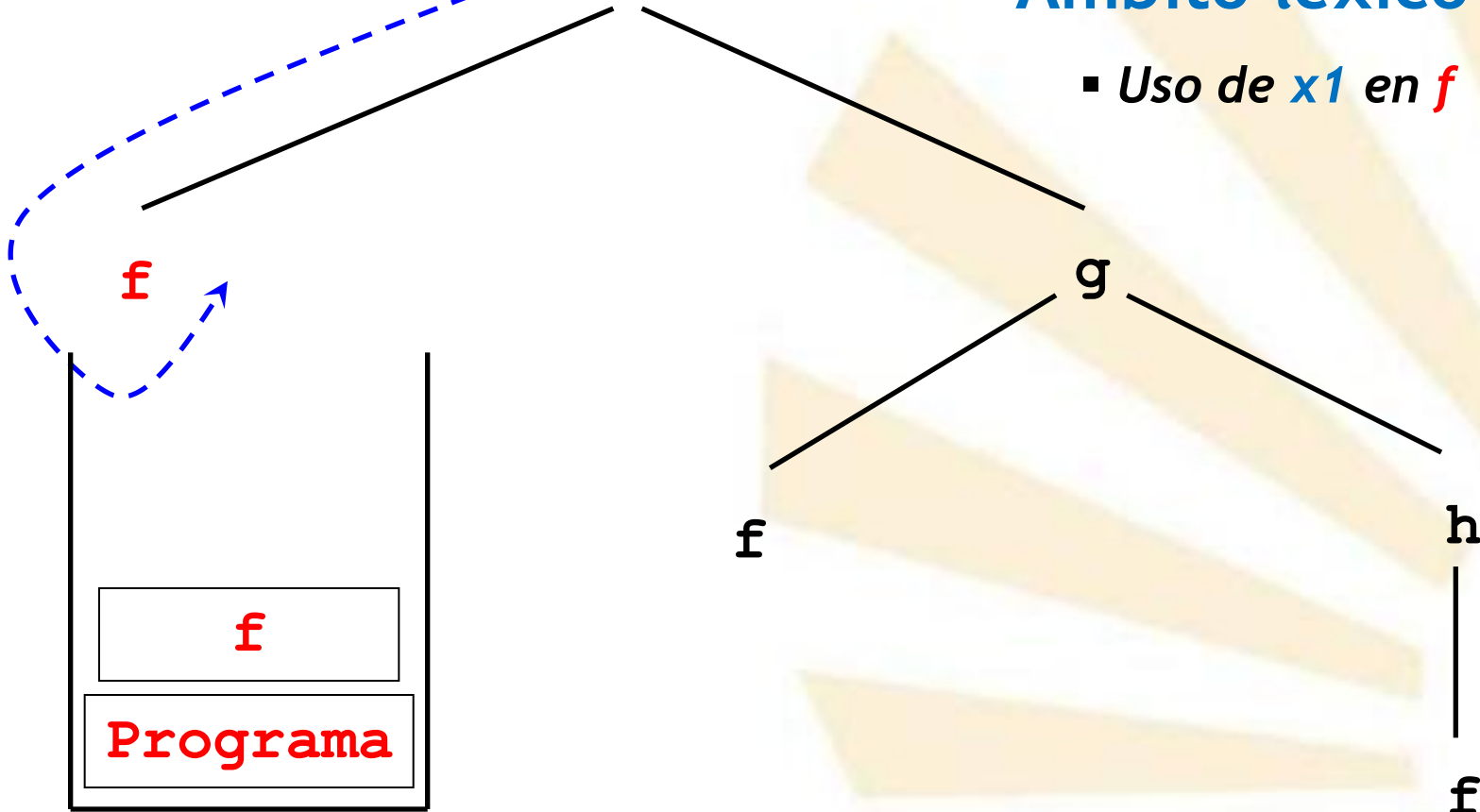
Llamada a **f** ←

Llamada a **g**

Programa

Ámbito léxico

- *Uso de x1 en f*

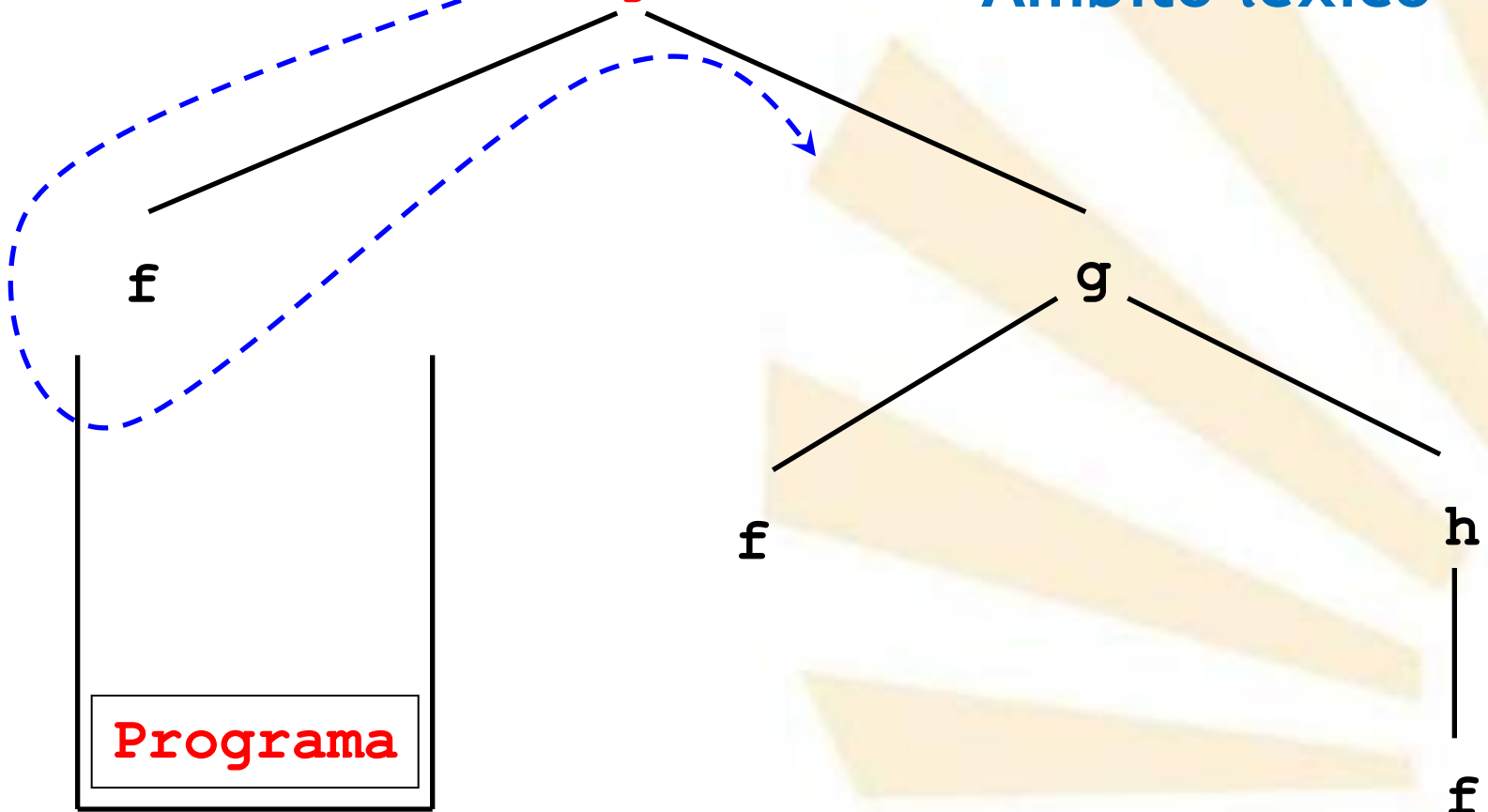


Pila de Activación

Árbol de Activación

Programa

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**

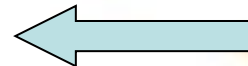
si no Uso de **x**

fin si

Uso de **x**

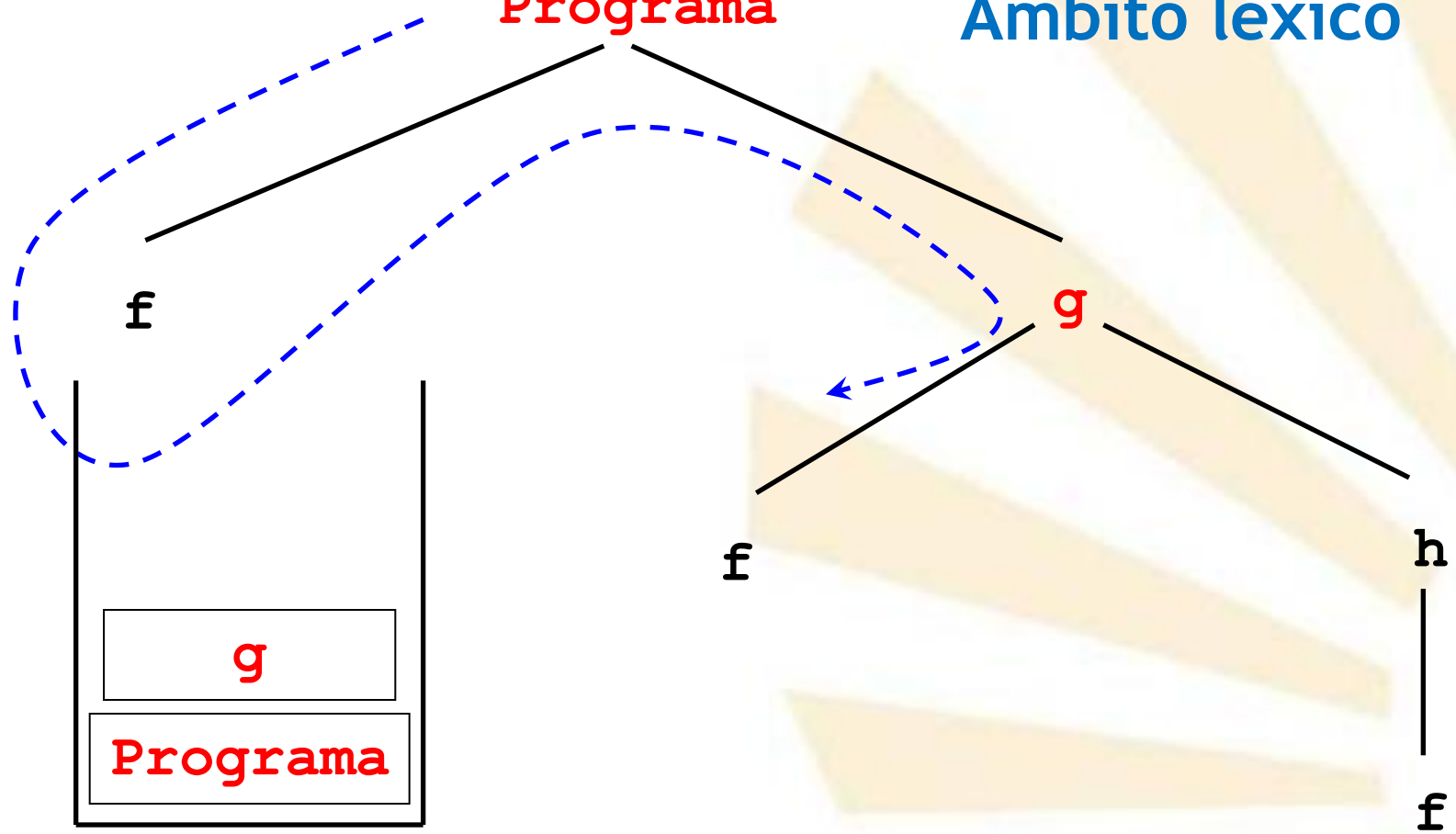
Llamada a **f**

Llamada a **g**



Programa

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

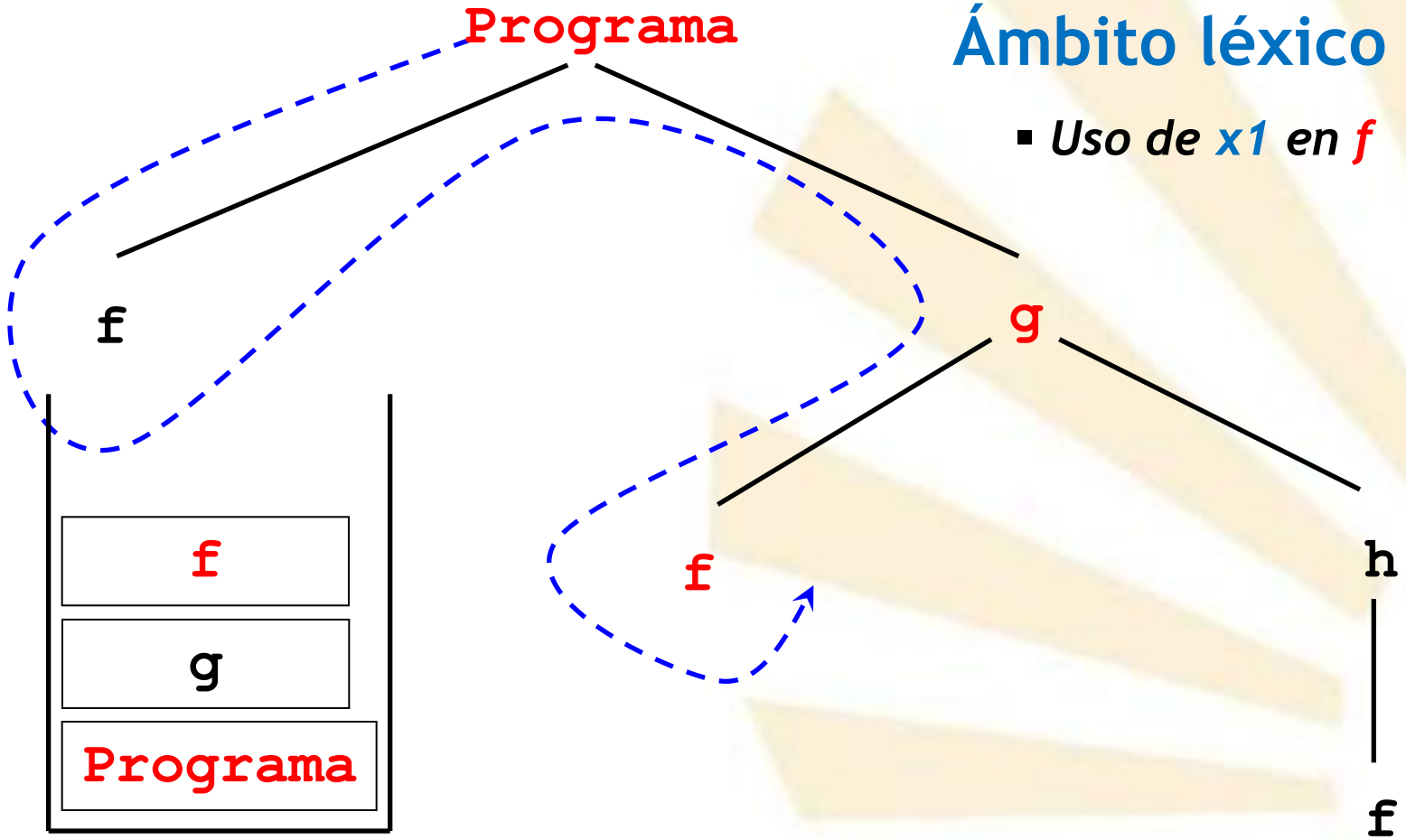
Uso de **x**

Llamada a **f**

Llamada a **g** ←

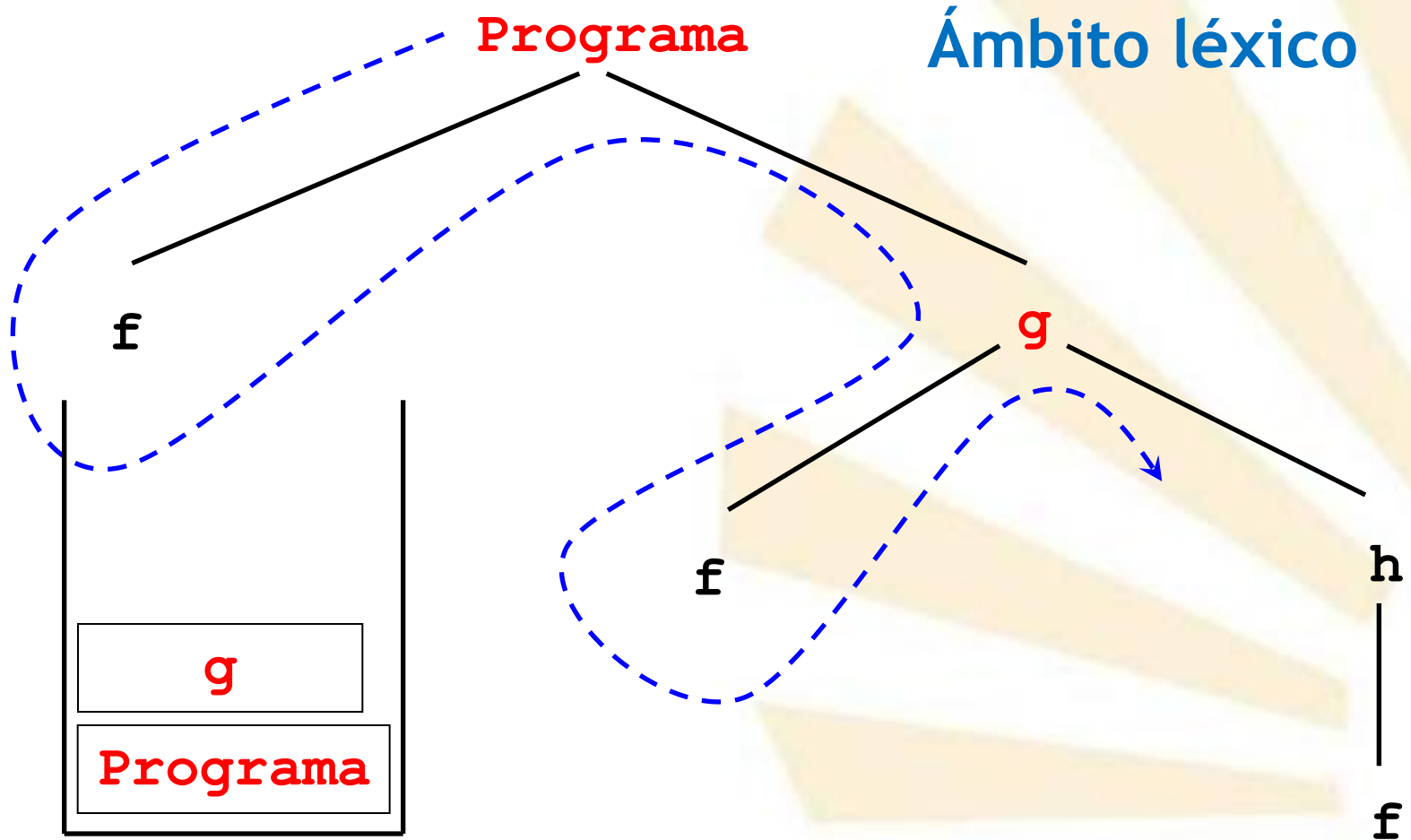
Ámbito léxico

- *Uso de x1 en f*



Pila de Activación

Árbol de Activación



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**

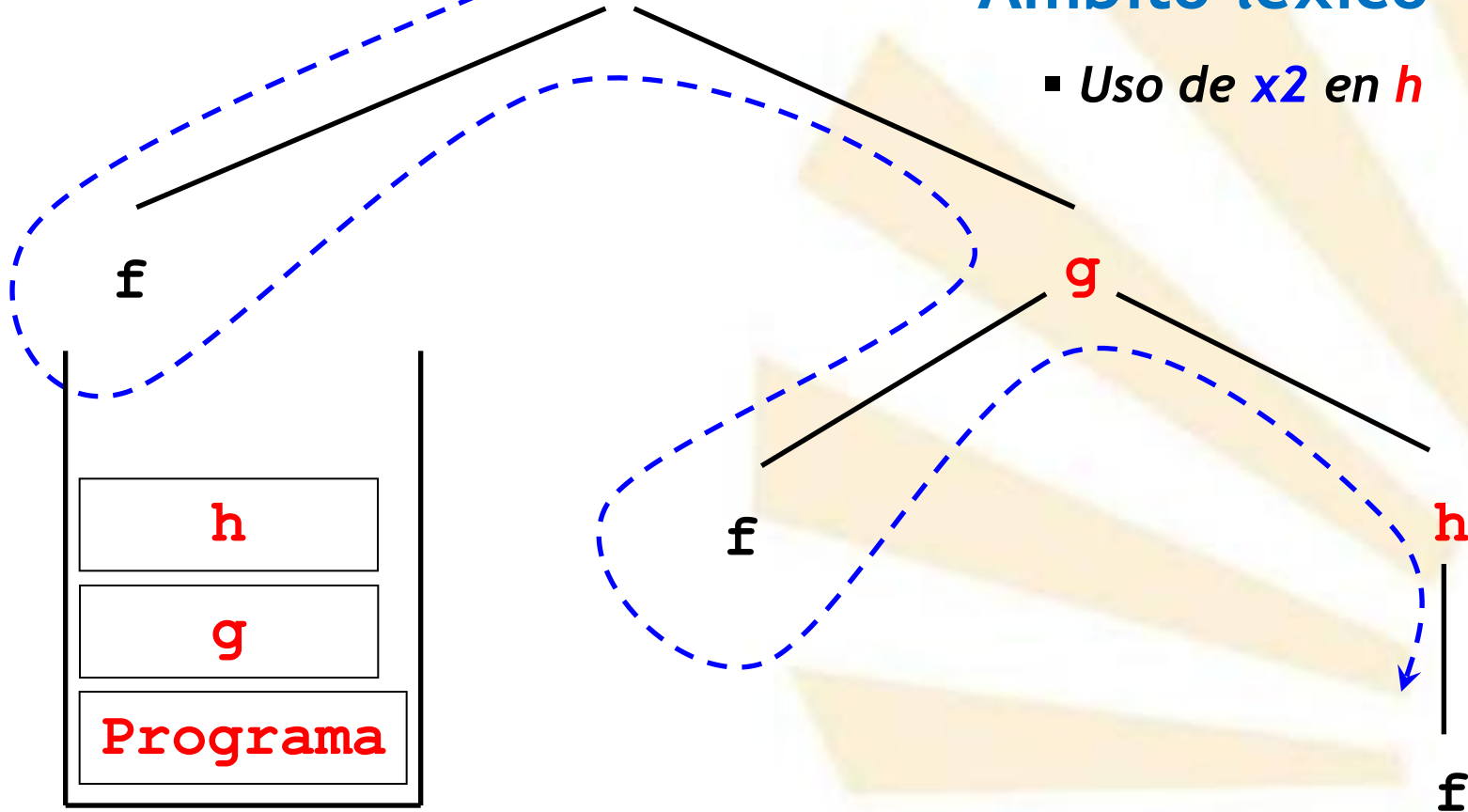
Llamada a **f**

Llamada a **g**

Programa

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

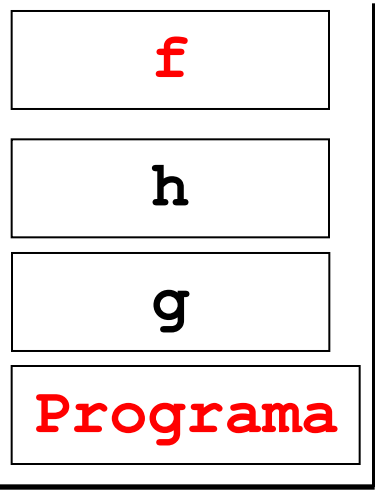
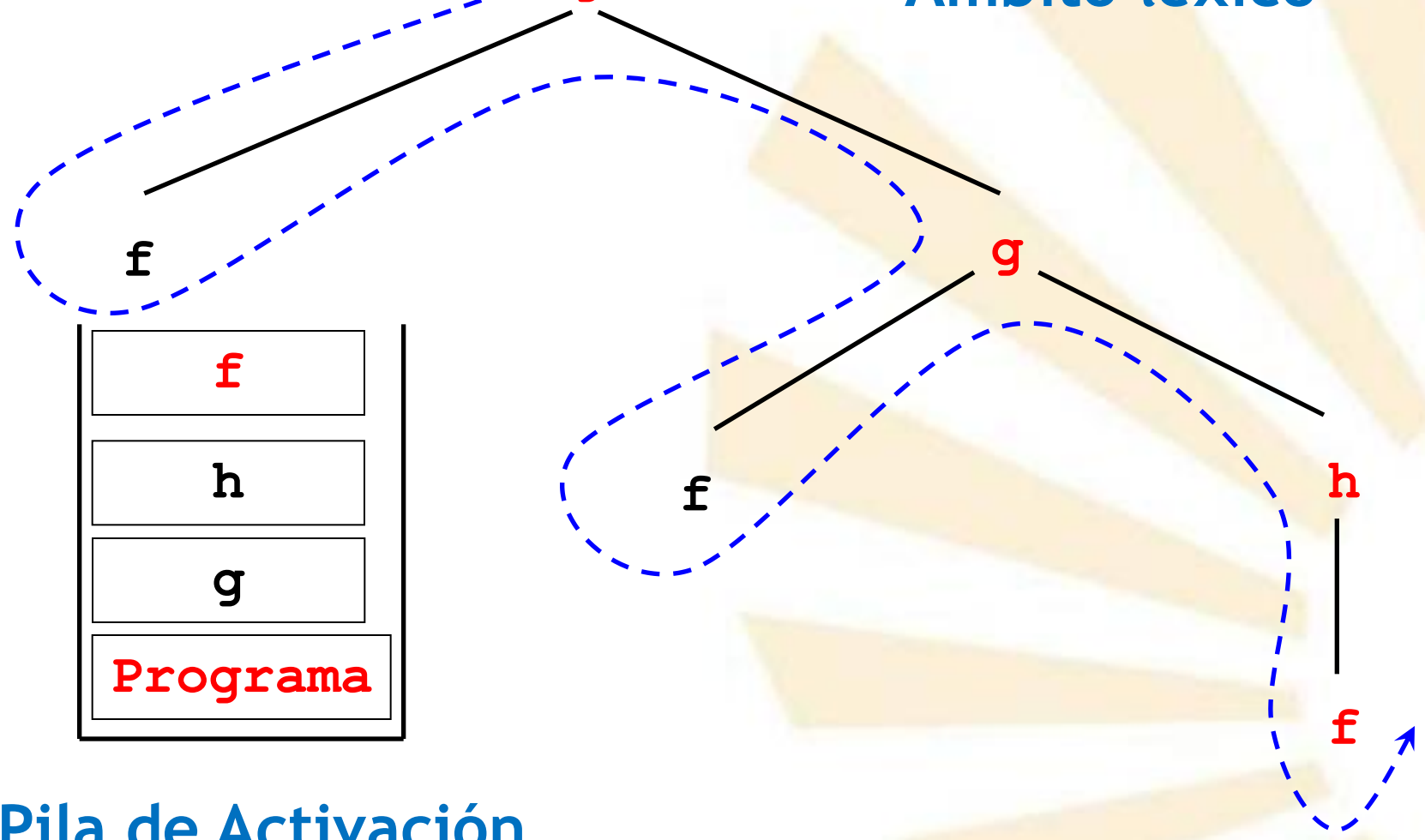
Uso de **x**

Llamada a **f**

Llamada a **g** ←

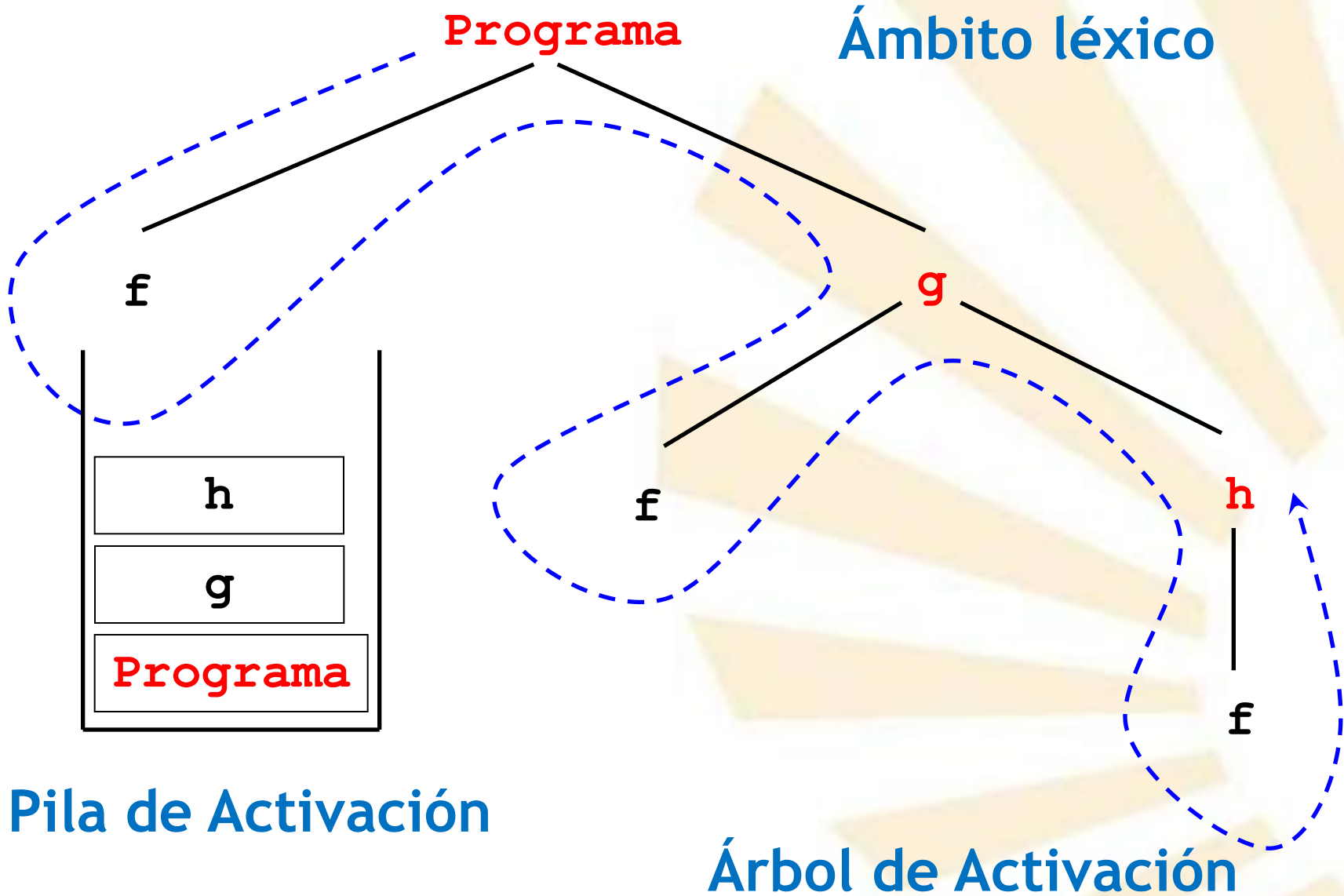
Programa

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

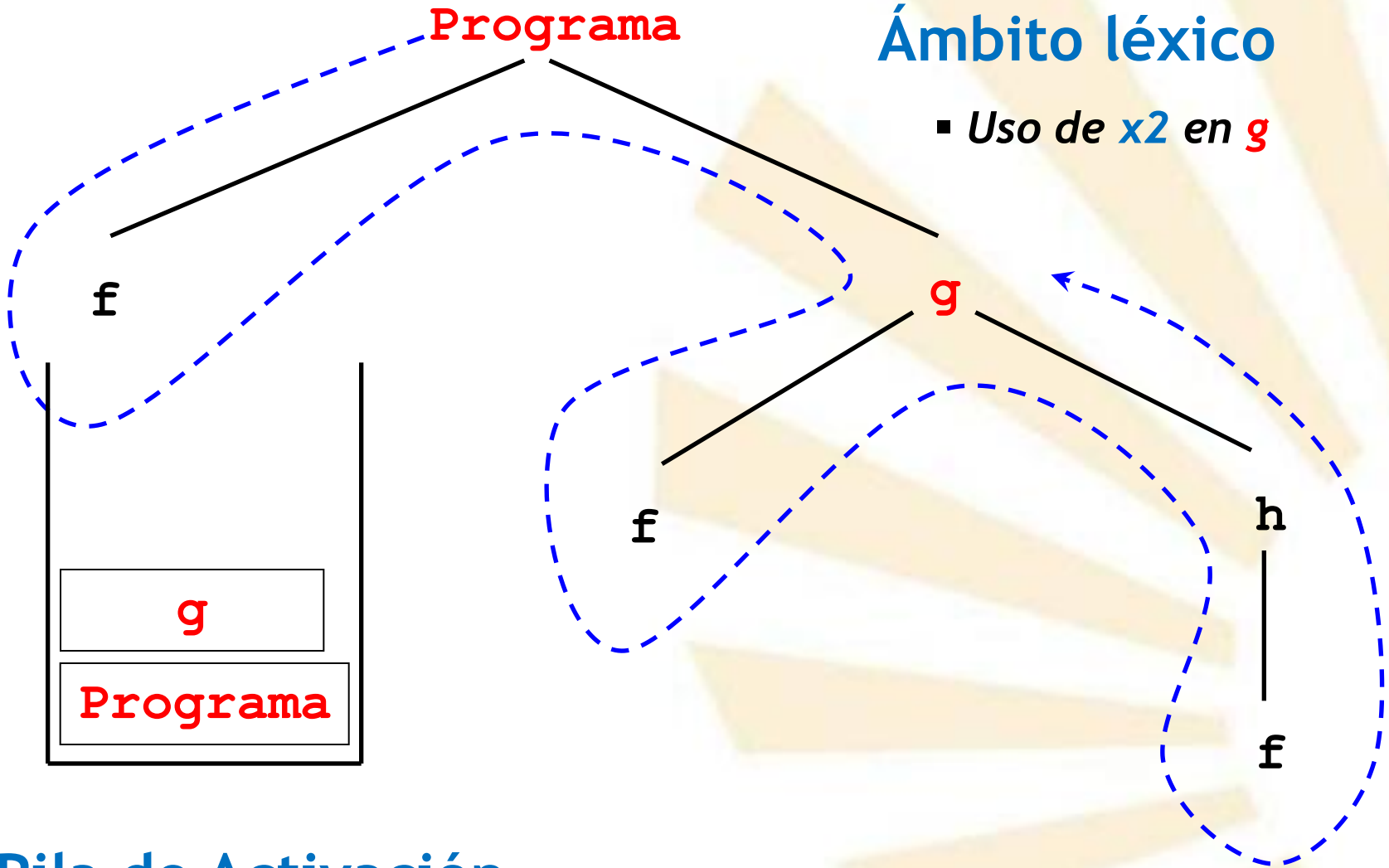
Uso de **x**

Llamada a **f**

Llamada a **g**

Ámbito léxico

- *Uso de x2 en g*



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**

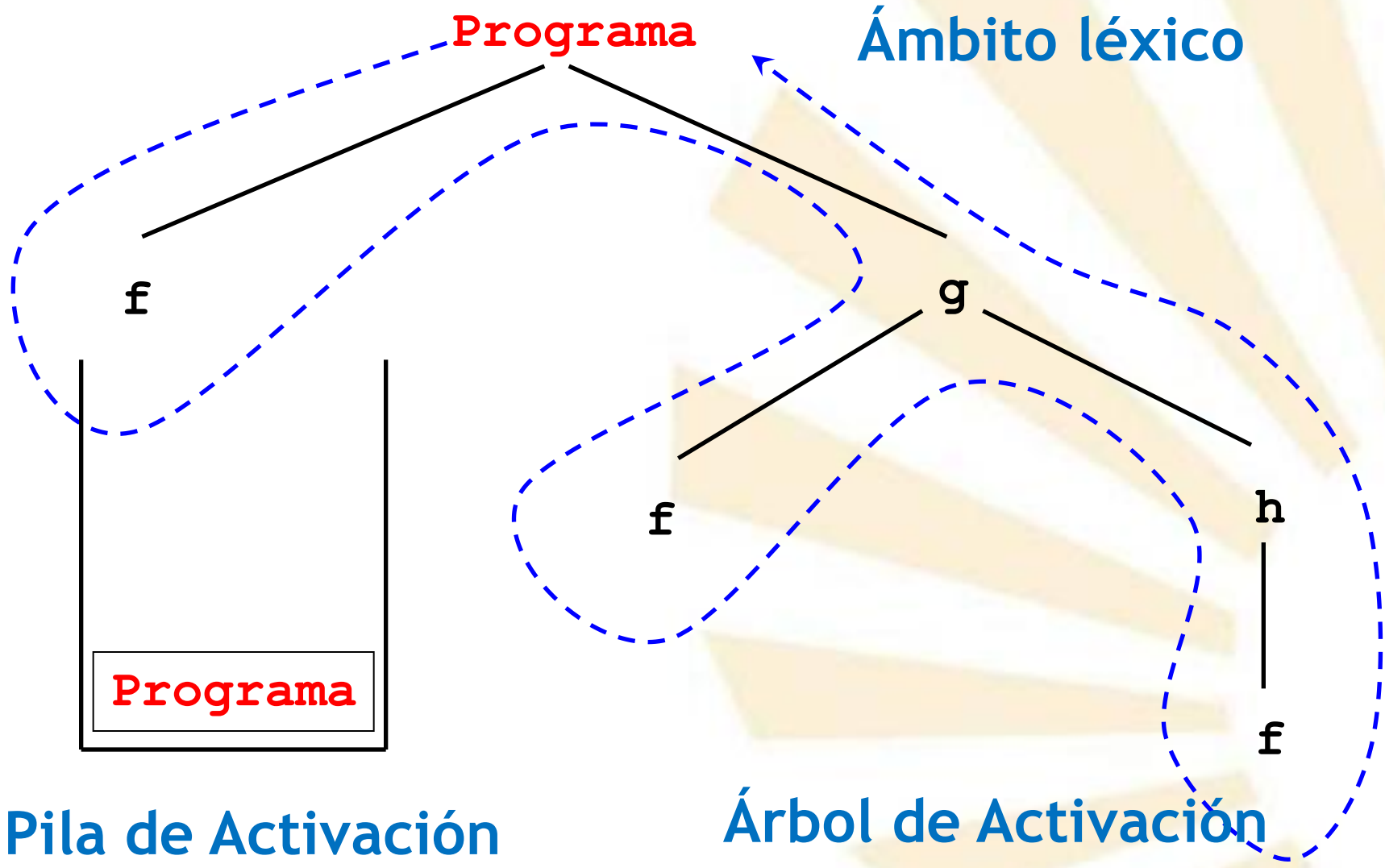
si no Uso de **x**

fin si

Uso de **x**

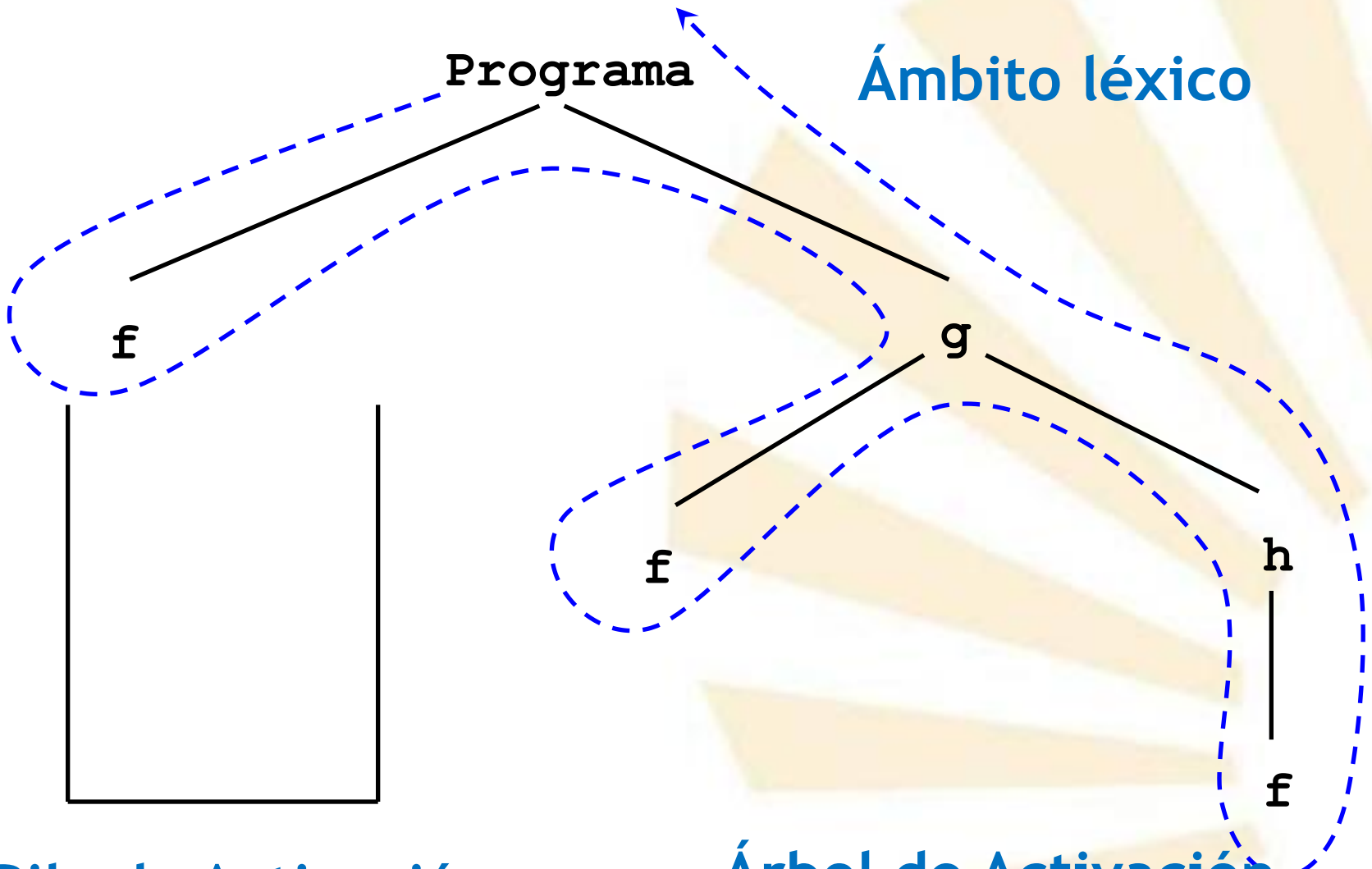
Llamada a **f**

Llamada a **g**



Programa

Ámbito léxico



f

g

f

h

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**

Ámbito dinámico

Programa

f

g

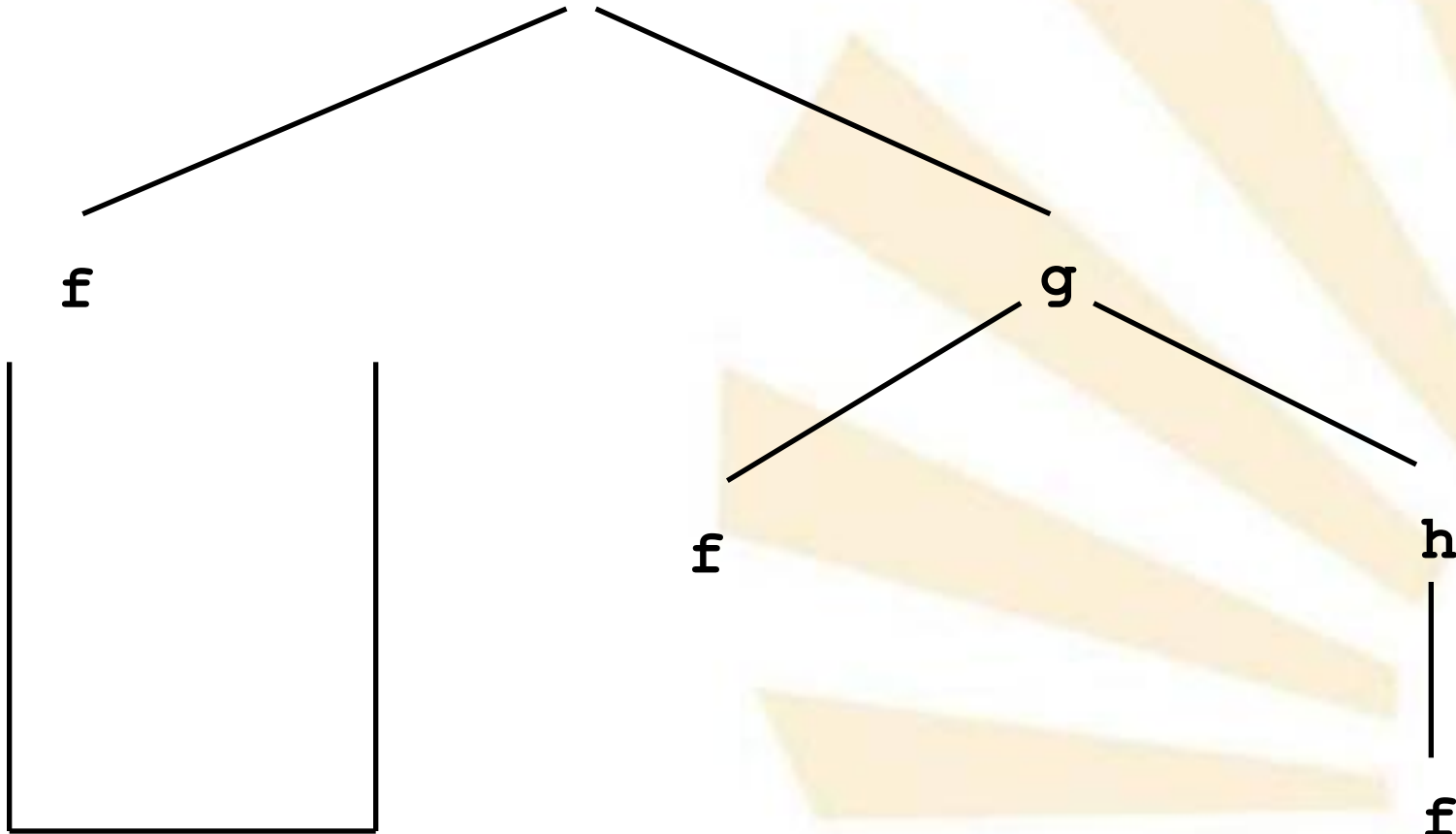
f

h

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**

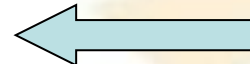
si no Uso de **x**

fin si

Uso de **x**

Llamada a **f**

Llamada a **g**



Programa

Ámbito dinámico

- *Uso de x1 en programa*

f

g

f

h

f

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**

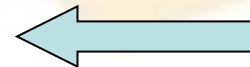
si no Uso de **x**

fin si

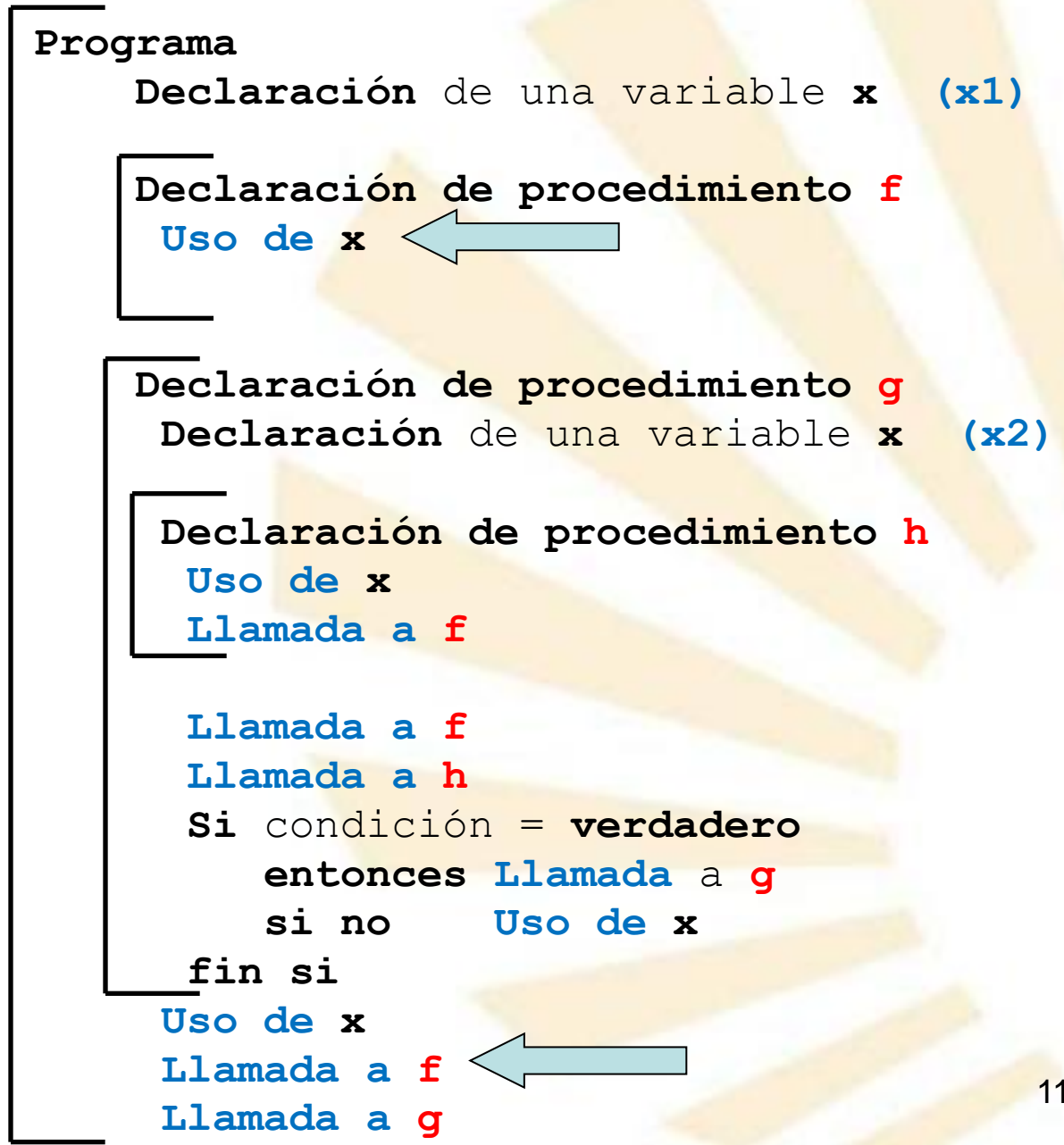
Uso de **x**

Llamada a **f**

Llamada a **g**



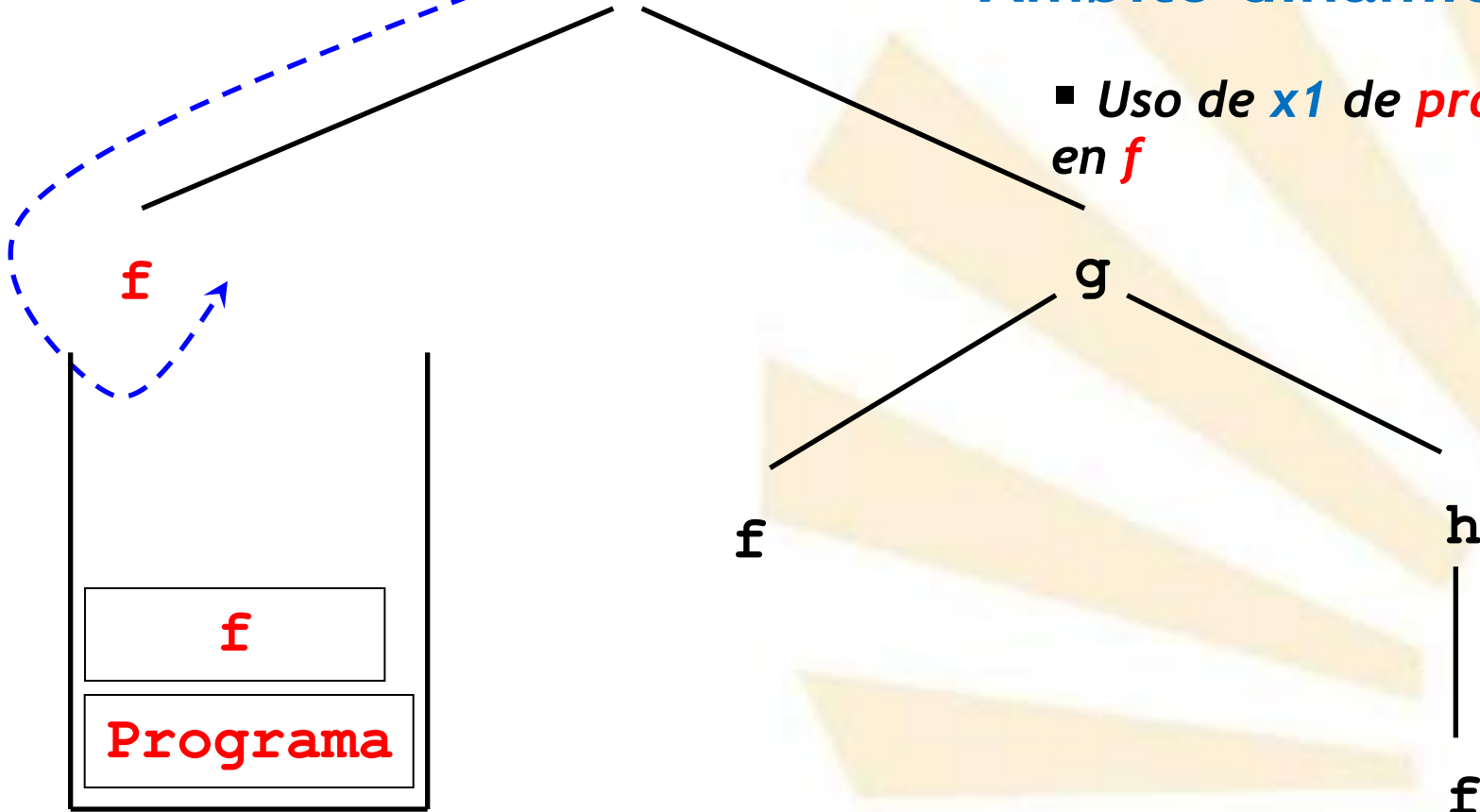
Ejecución con ámbito dinámico



Programa

Ámbito dinámico

- *Uso de x1 de programa en f*

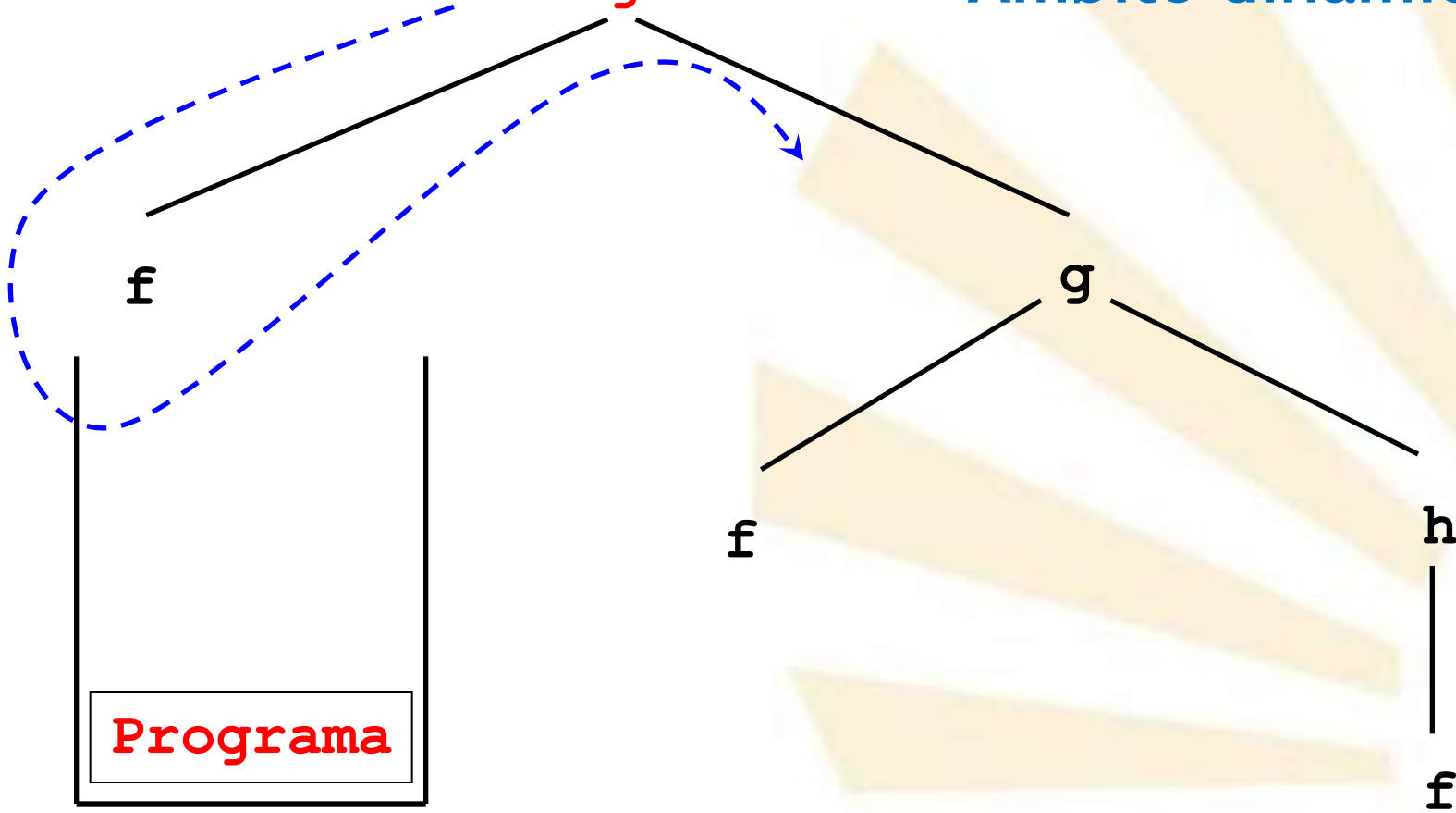


Pila de Activación

Árbol de Activación

Programa

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**

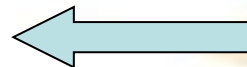
si no Uso de **x**

fin si

Uso de **x**

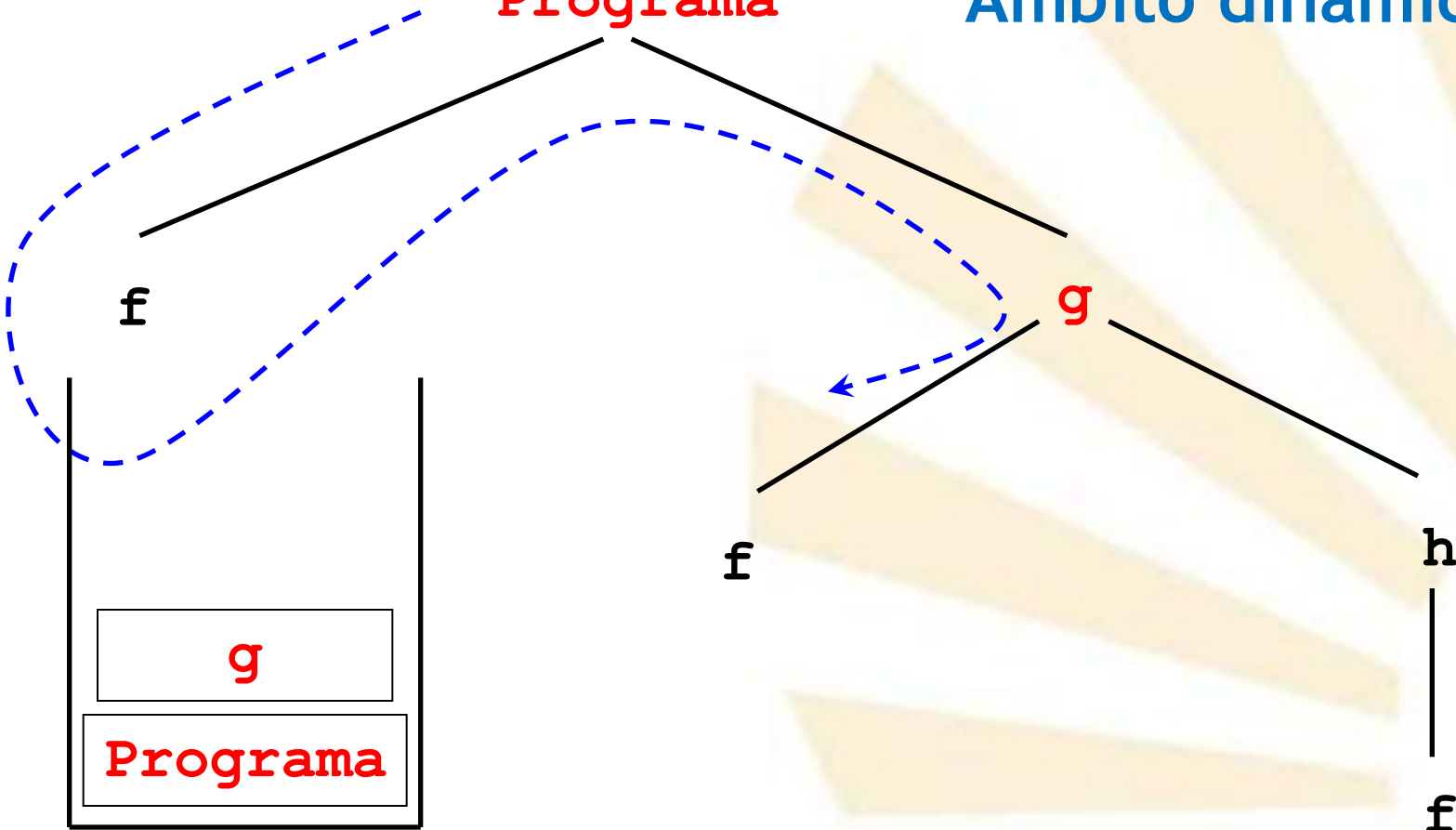
Llamada a **f**

Llamada a **g**



Programa

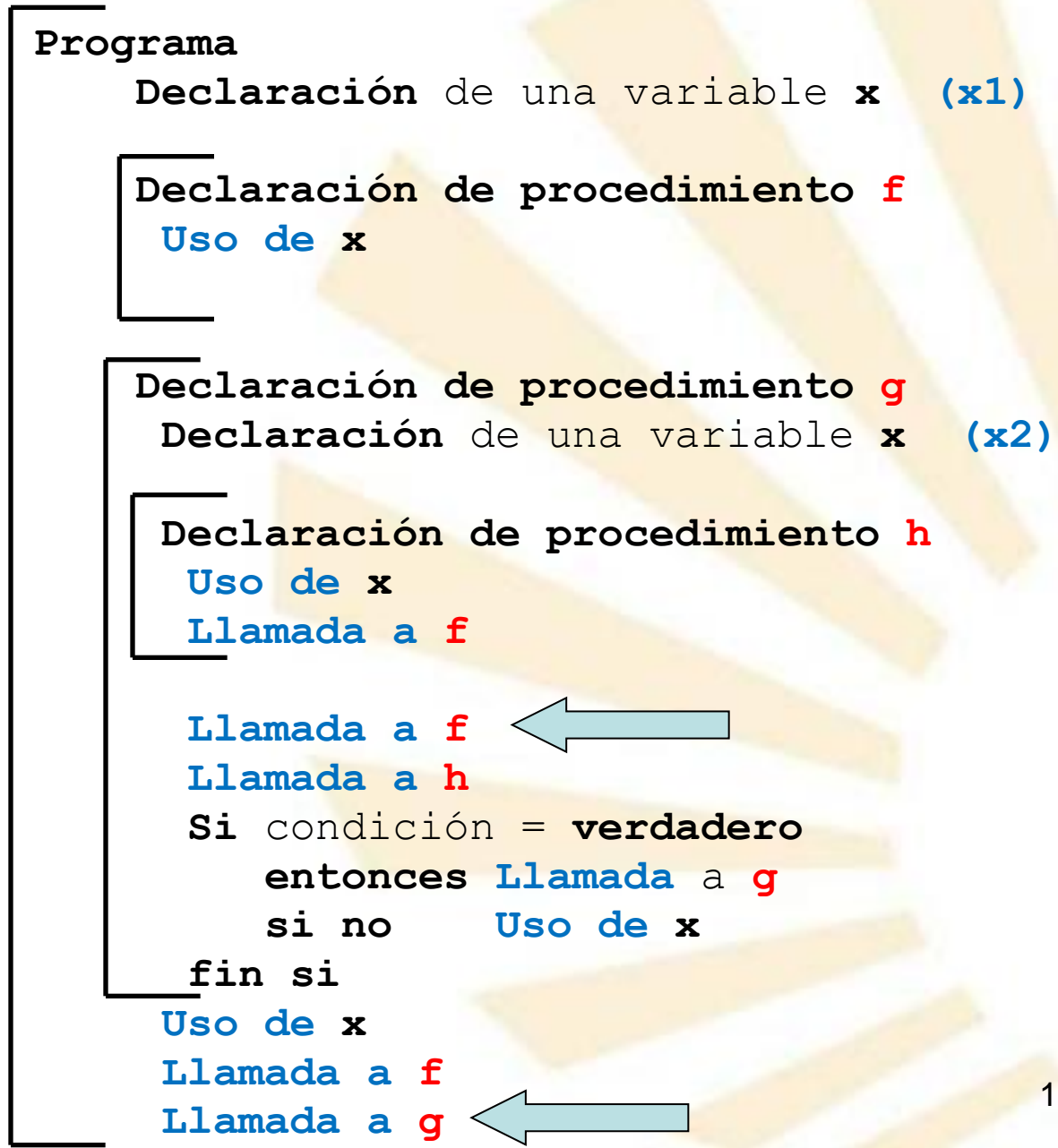
Ámbito dinámico



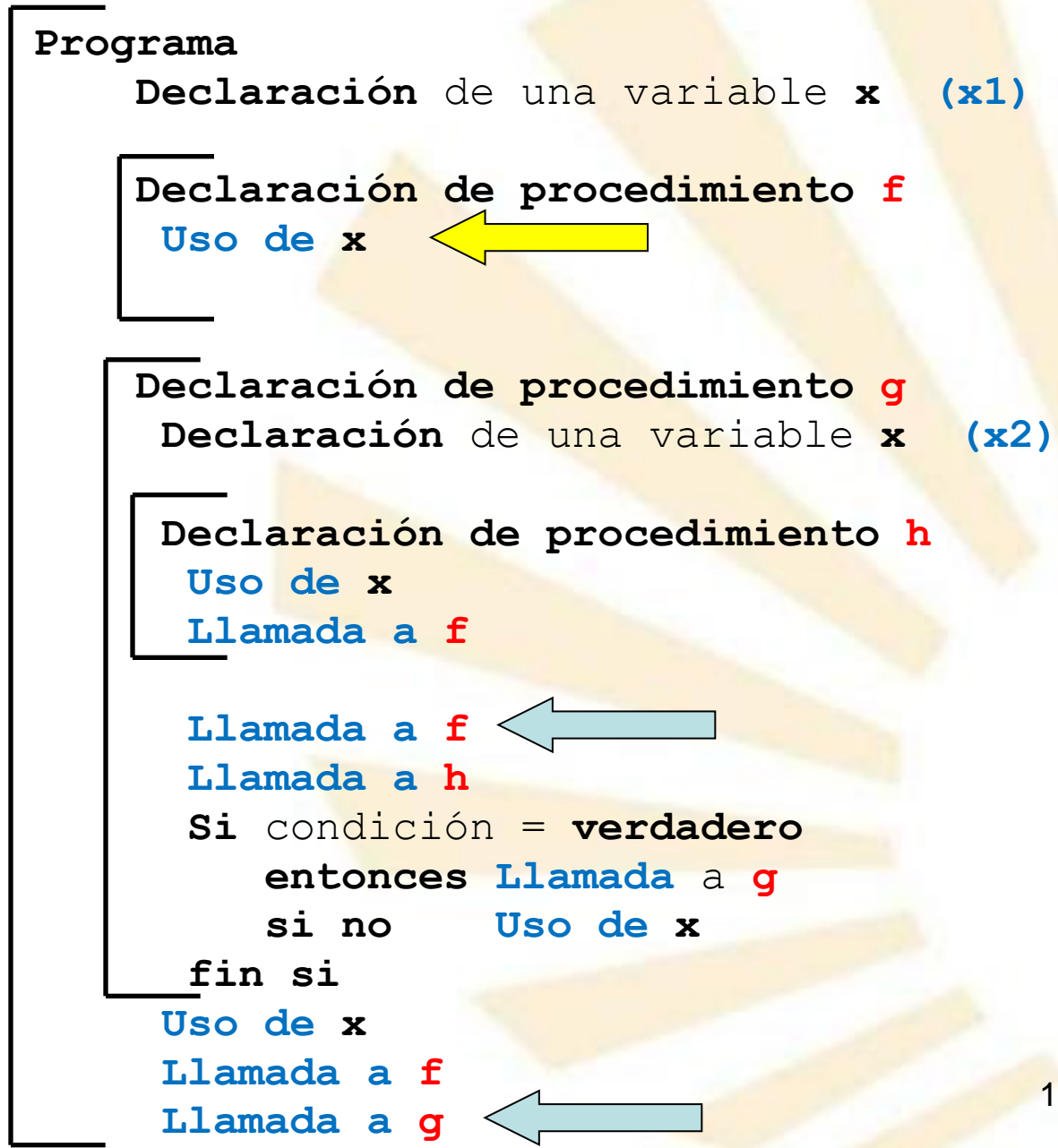
Pila de Activación

Árbol de Activación

Ejecución con ámbito dinámico

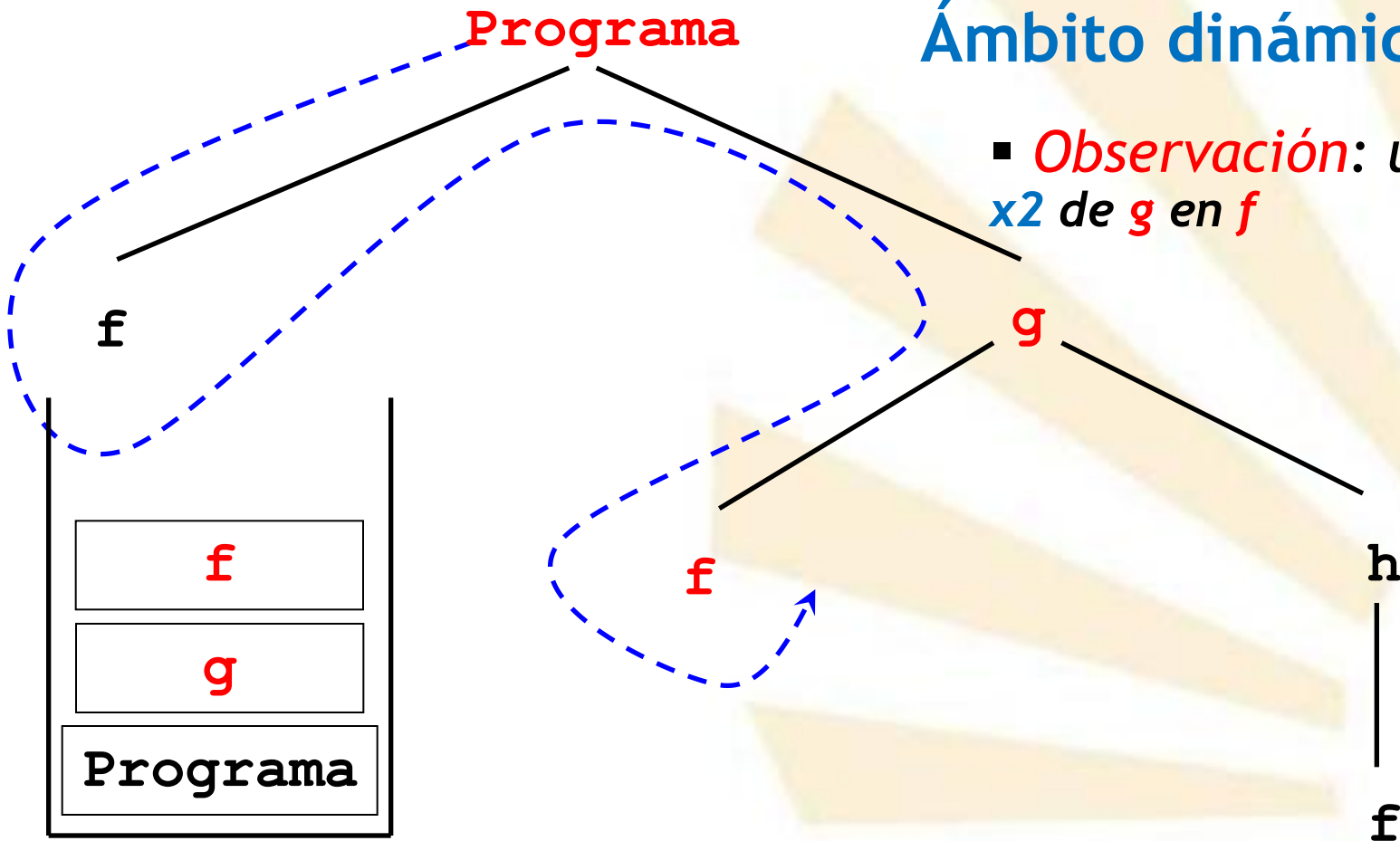


Ejecución con ámbito dinámico

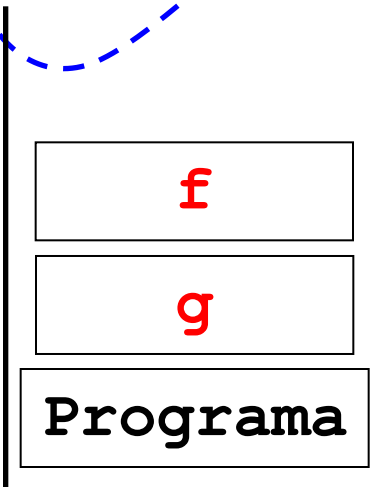


Ámbito dinámico

- *Observación*: Uso de x_2 de g en f



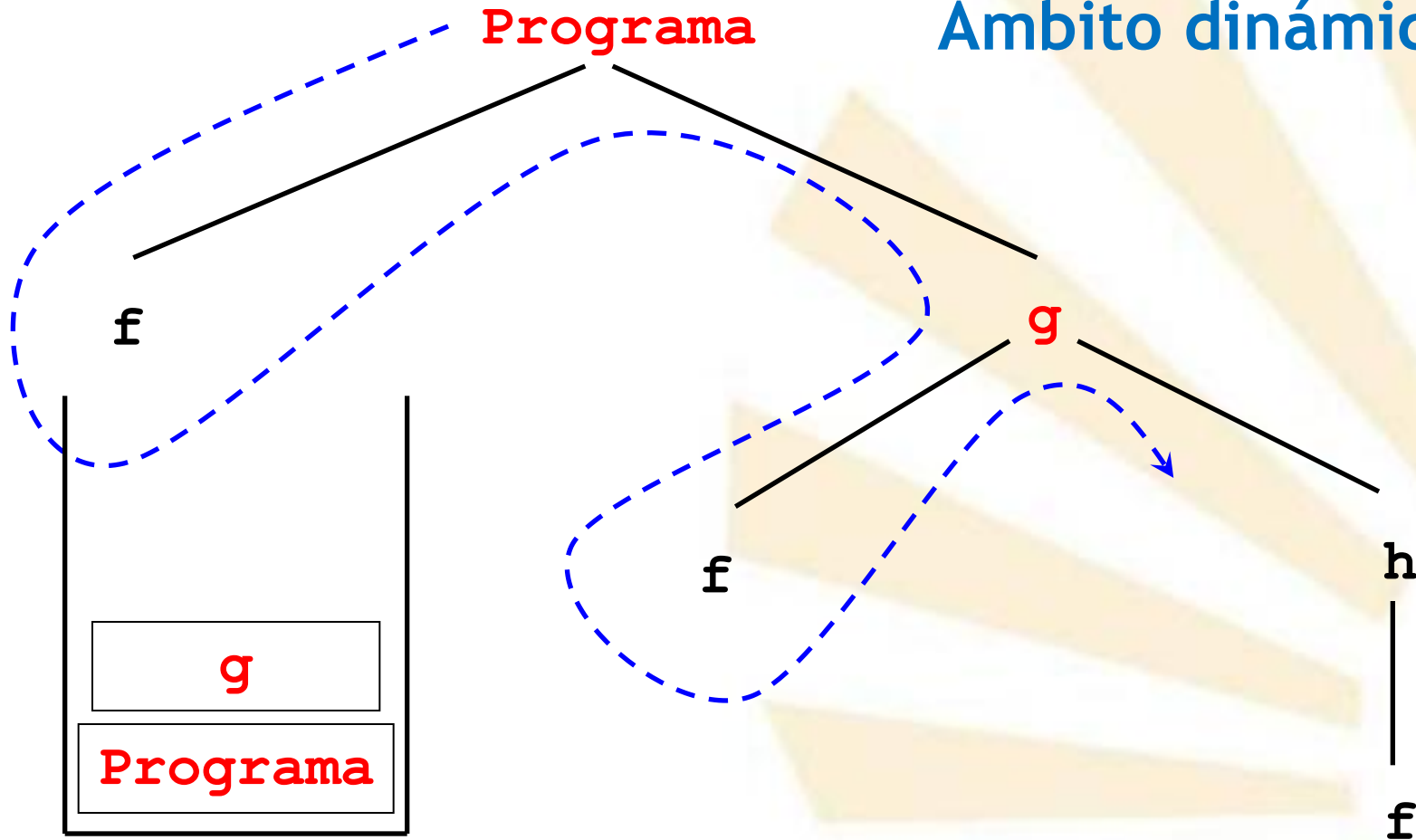
Pila de Activación



Árbol de Activación

Ámbito dinámico

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**

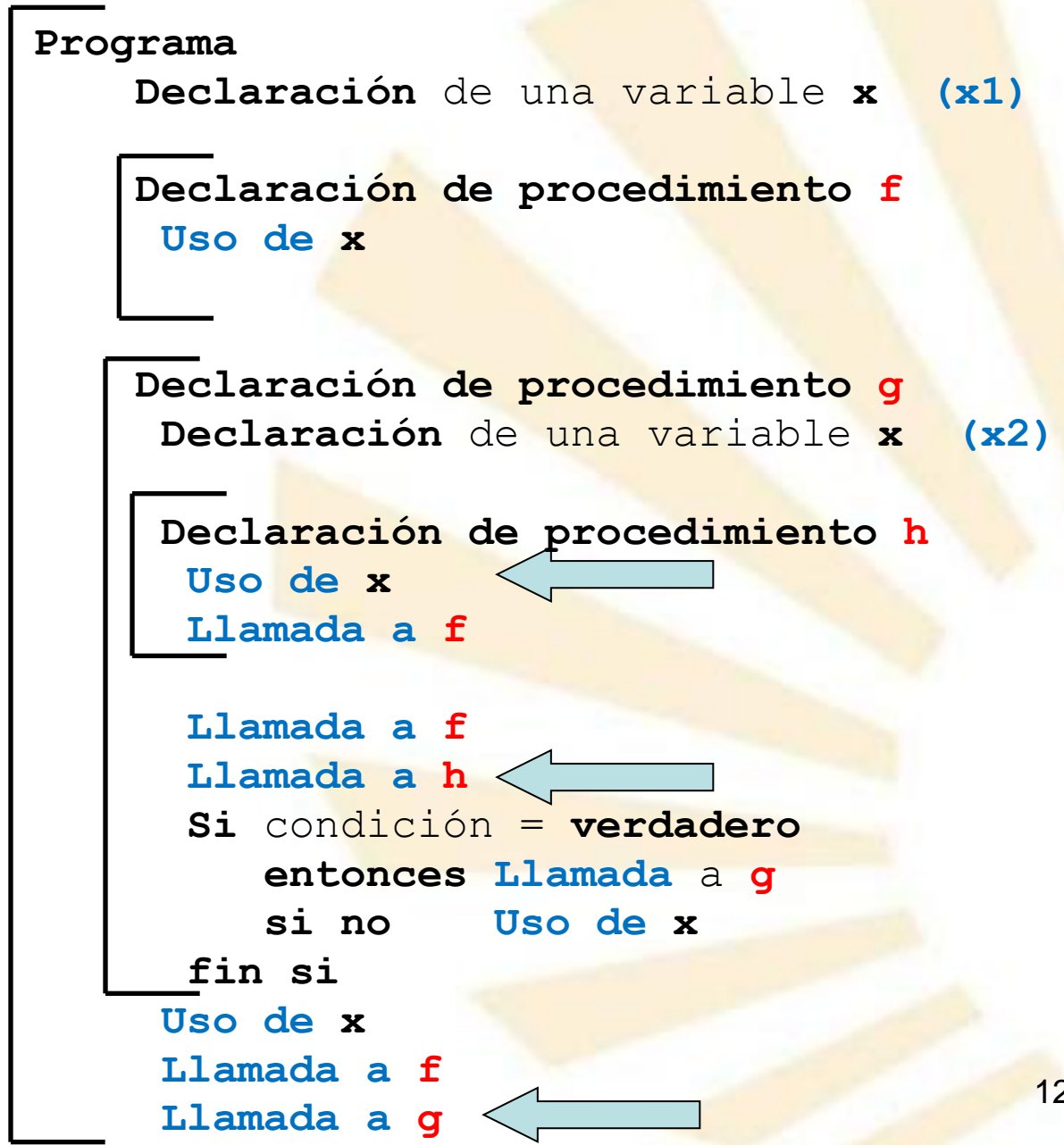
fin si

Uso de **x**

Llamada a **f**

Llamada a **g**

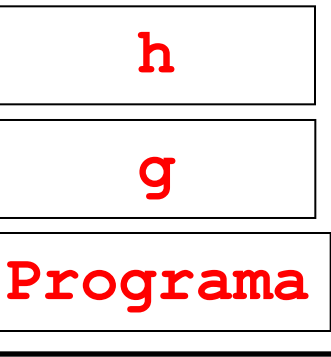
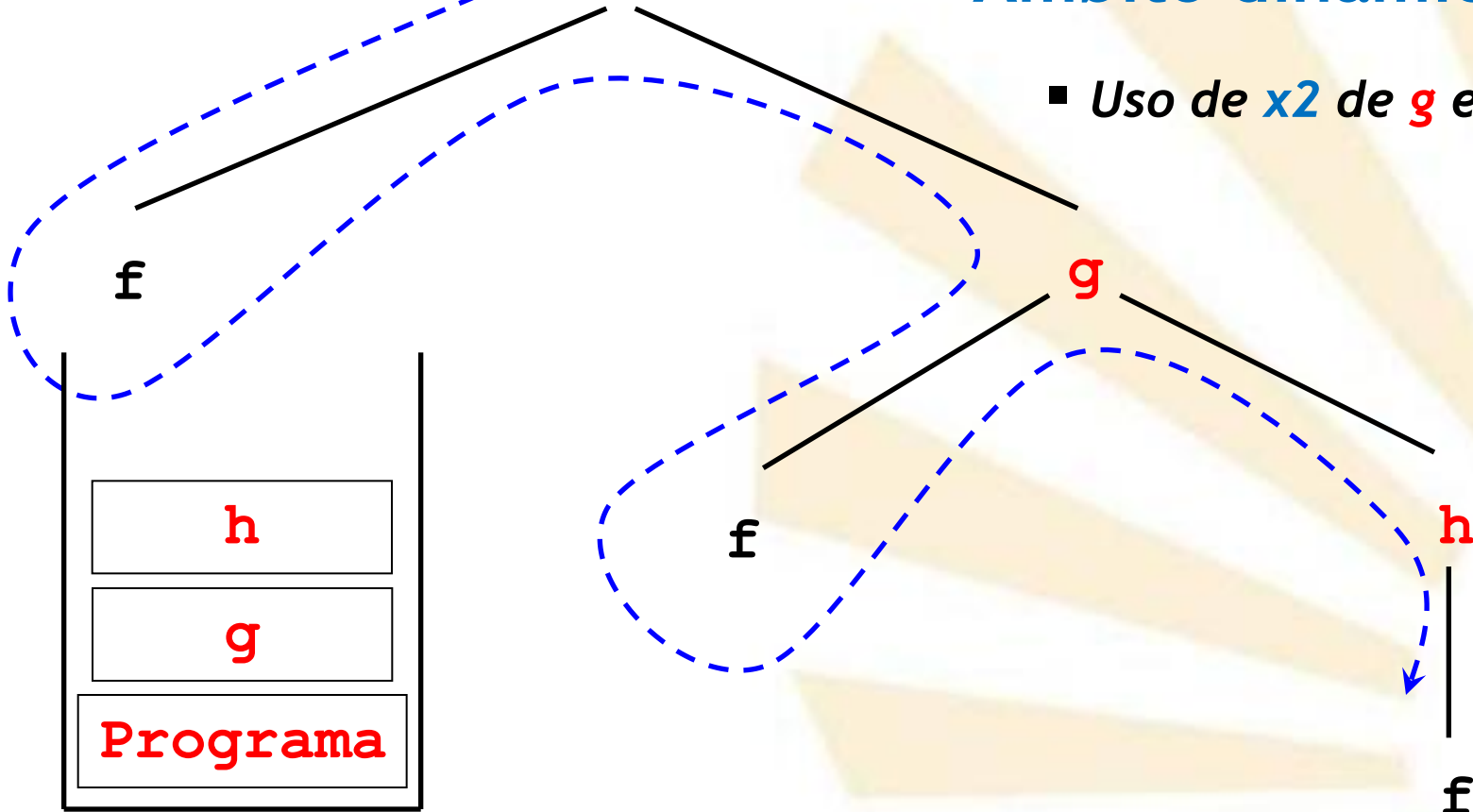
Ejecución con
ámbito
dinámico



Programa

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**

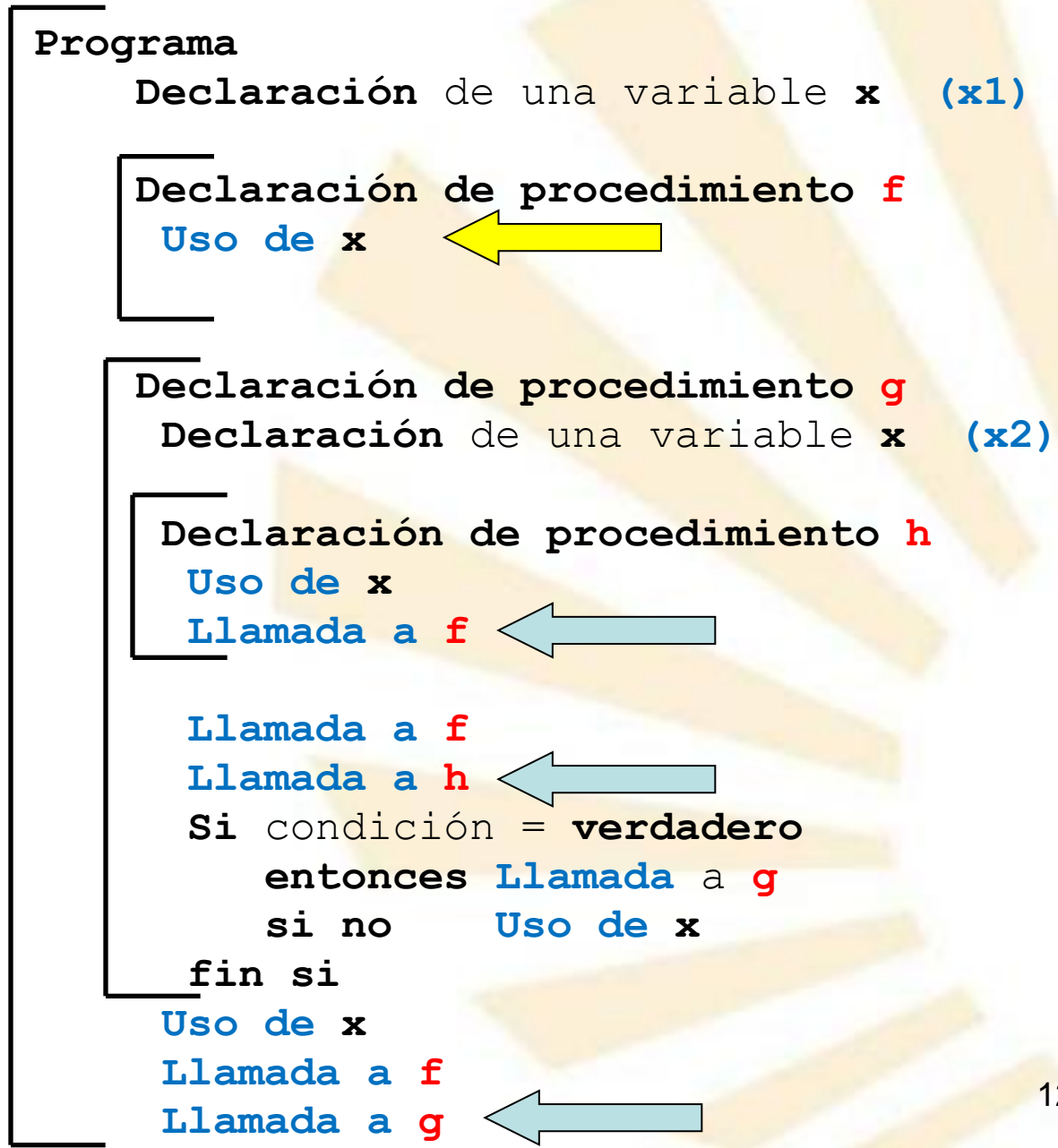
fin si

Uso de **x**

Llamada a **f**

Llamada a **g**

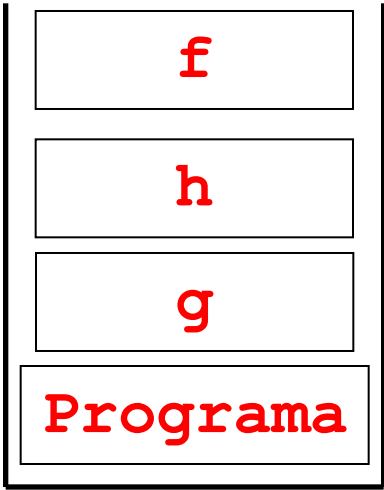
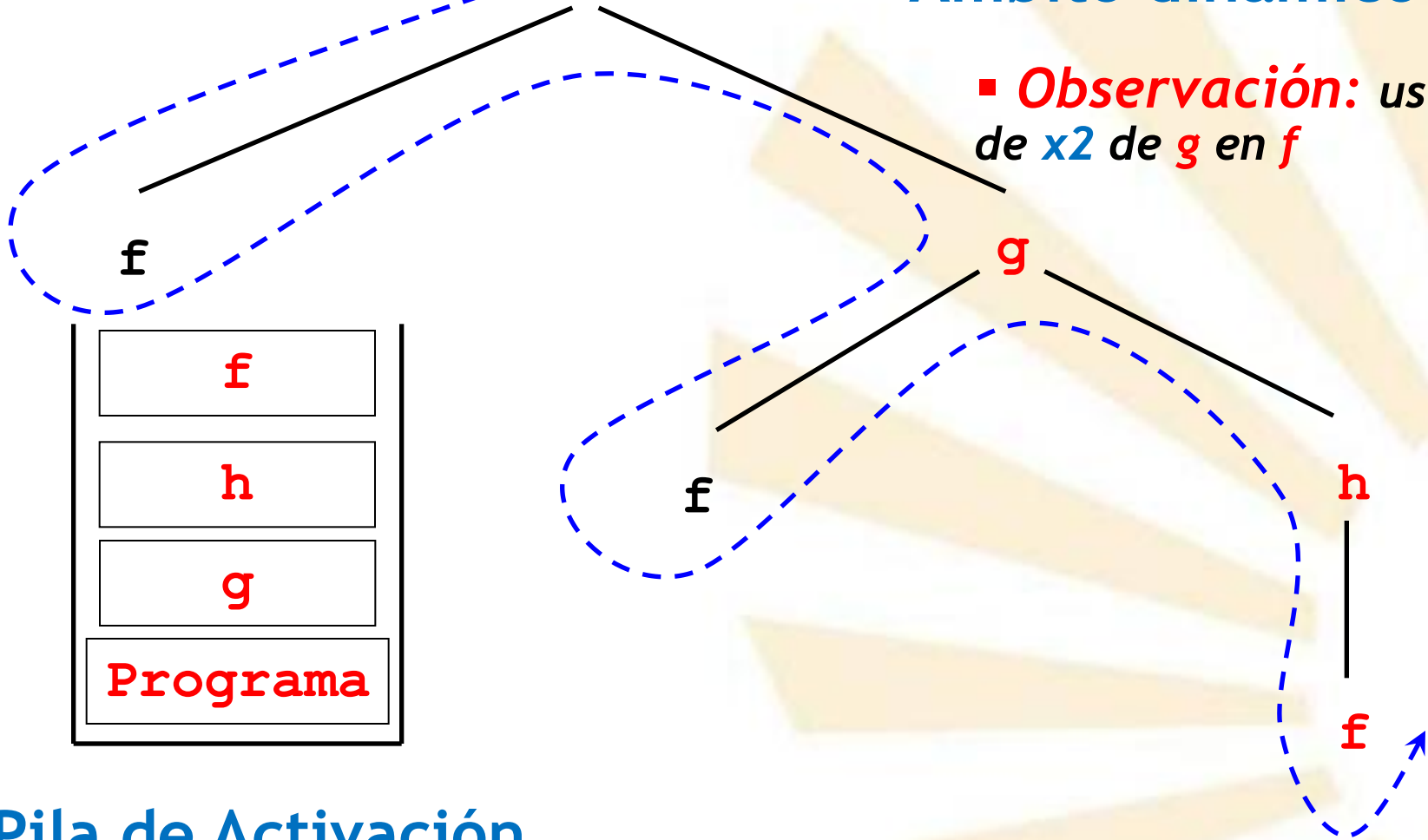
Ejecución con
ámbito
dinámico



Programa

Ámbito dinámico

▪ **Observación:** uso de x_2 de g en f

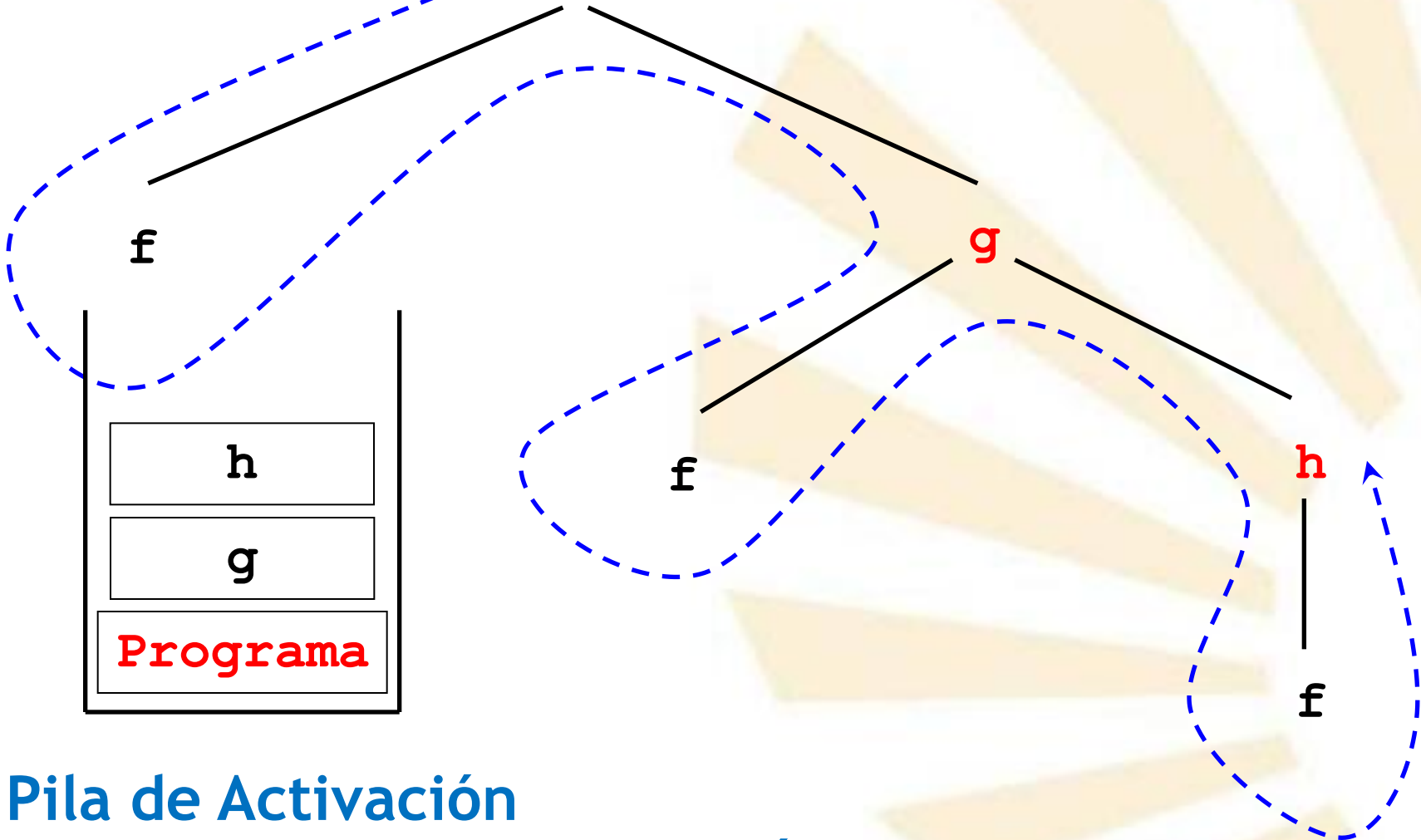


Pila de Activación

Árbol de Activación

Programa

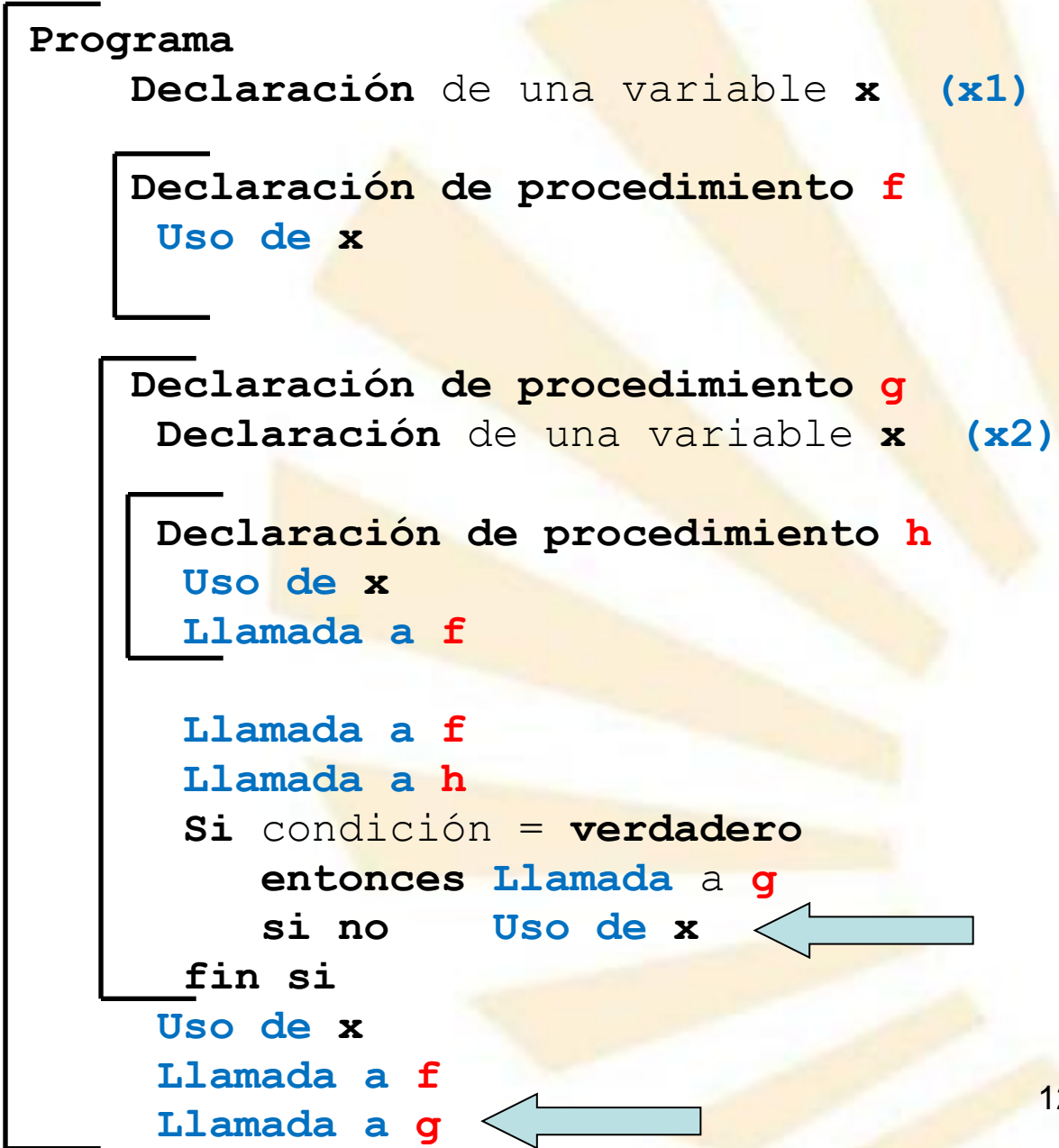
Ámbito dinámico



Pila de Activación

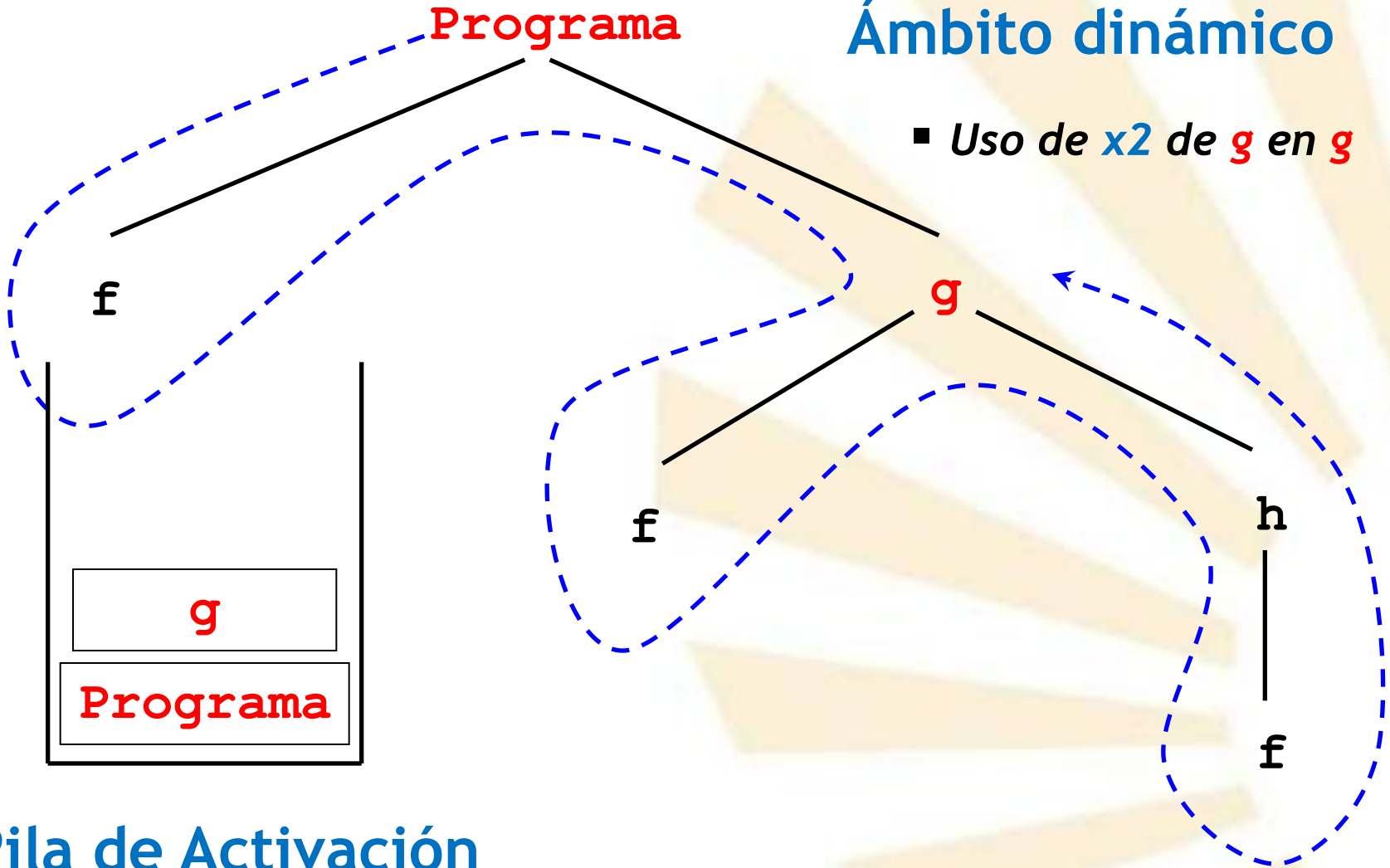
Árbol de Activación

Ejecución con
ámbito
dinámico



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**

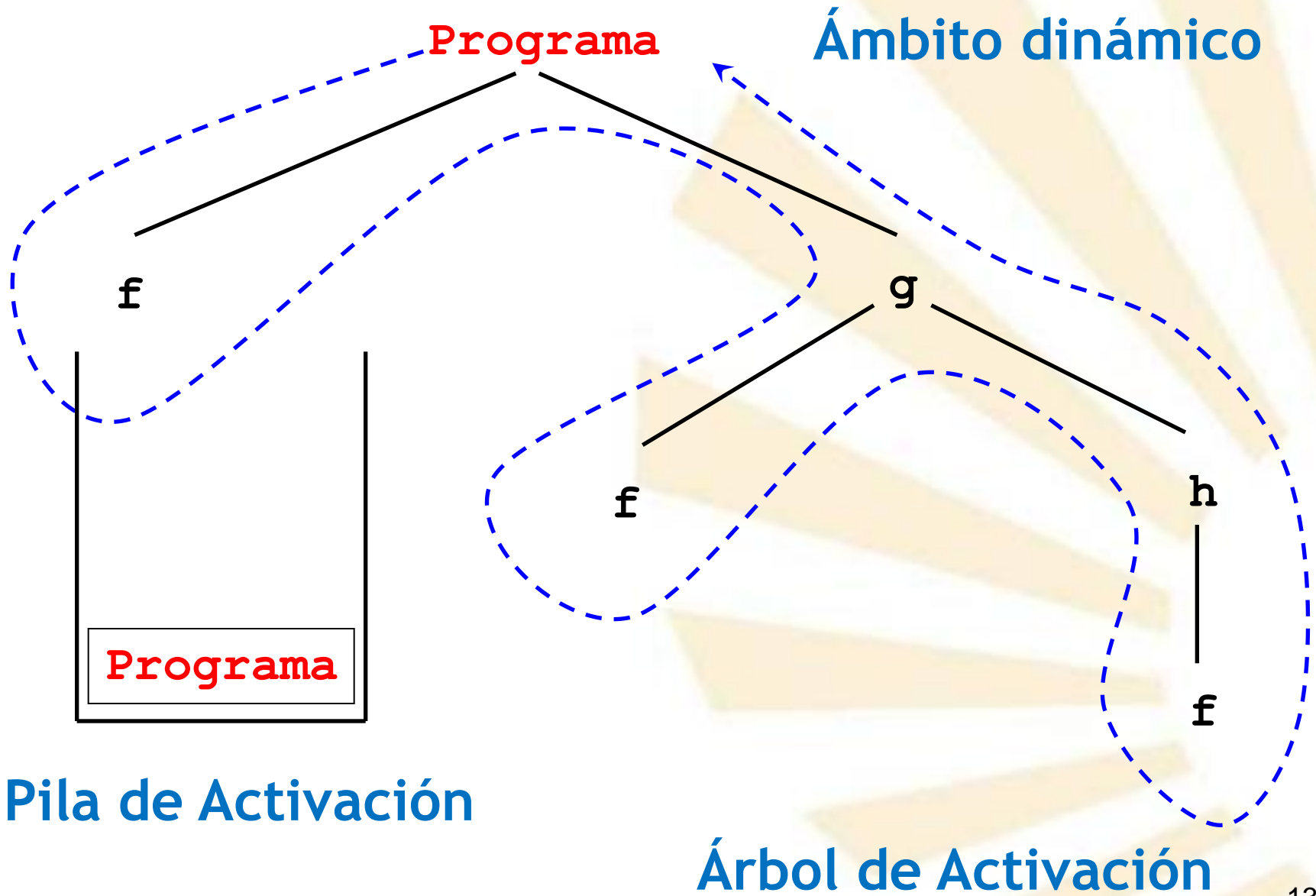
si no Uso de **x**

fin si

Uso de **x**

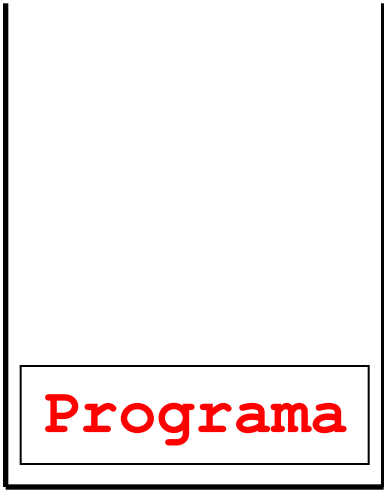
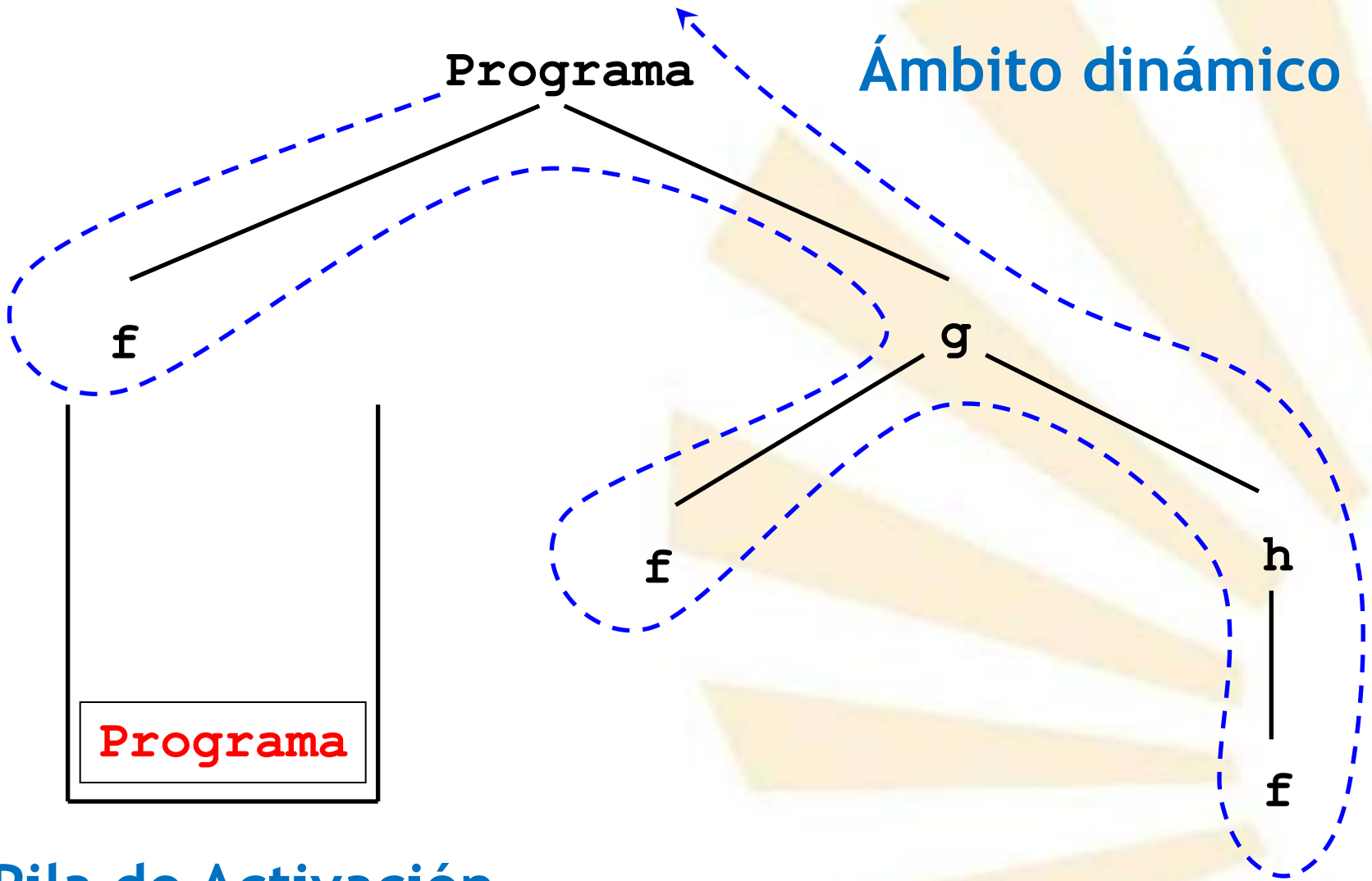
Llamada a **f**

Llamada a **g**



Programa

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 Common LISP:** reglas de **ámbito léxico**
- ***Revised⁵ Report on the Algorithmic Language Scheme***

2. Reseña Histórica de Scheme

- **Origen de Scheme**

- **Estructura de los programas de Scheme**

- **Secuencia de**

- **definiciones de funciones y variables**

- **y expresiones**



UNIVERSIDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA

DEPARTAMENTO DE
INFORMÁTICA Y ANÁLISIS NUMÉRICO



PROGRAMACIÓN DECLARATIVA

INGENIERÍA INFORMÁTICA

CUARTO CURSO

PRIMER CUATRIMESTRE



Tema 1.- Introducción al Lenguaje Scheme