Programar función Ackermann - Threads

MFC – Modelos Formales de Computación

MITSS

Sergi Sanz Carreres

Índice

[Función de Ackermann 2](#_Toc25922330)

[Ejemplo Recursivo: 2](#_Toc25922331)

[Ejemplo Concurrencia: 3](#_Toc25922332)

[Resultados obtenidos: 4](#_Toc25922333)

[Ejemplo 1: 4](#_Toc25922334)

[Ejemplo 2: 4](#_Toc25922335)

[Ejemplo 3: 5](#_Toc25922336)

[Ejemplo 4: 5](#_Toc25922337)

# Función de Ackermann

La función Ackermann es una función recursiva encontrada en 1926 por Wilhelm Ackermann, esta función toma dos números naturales como argumentos y devuelve un único número natural. Como norma general se define como sigue:

Imagen que contiene objeto

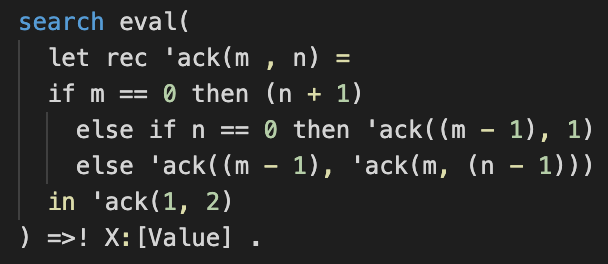
Descripción generada automáticamente

Una de las particularidades de la función Ackermann es que no cae en las consideradas funciones recursivas primitivas, debido principalmente a su desmesurado crecimiento.

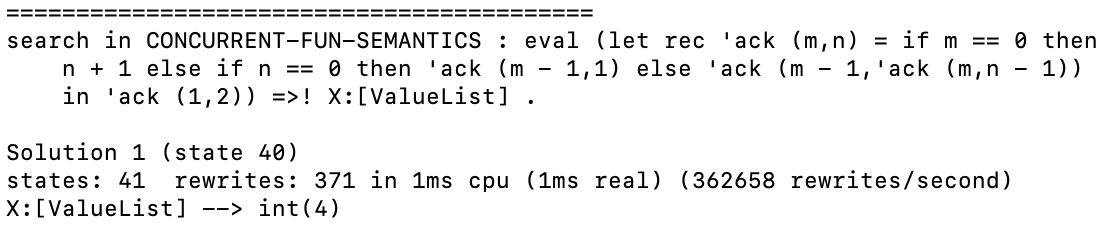
Este teorema se lleva a cabo mediante la técnica de reducción al absurdo, el cual es uno de los métodos lógicos de demostración más usado en matemáticas para demostrar la validez (o invalidez) de proposiciones categóricas.

# Ejemplo Recursivo:

Antes de cumplir con las necesidades requeridas para el desarrollo de este ejercicio haciendo uso de la ejecución de hilos, se ha procedido a desarrollar de forma recursiva esta función con el siguiente fragmento de código en Maude:



Dando el siguiente resultado:



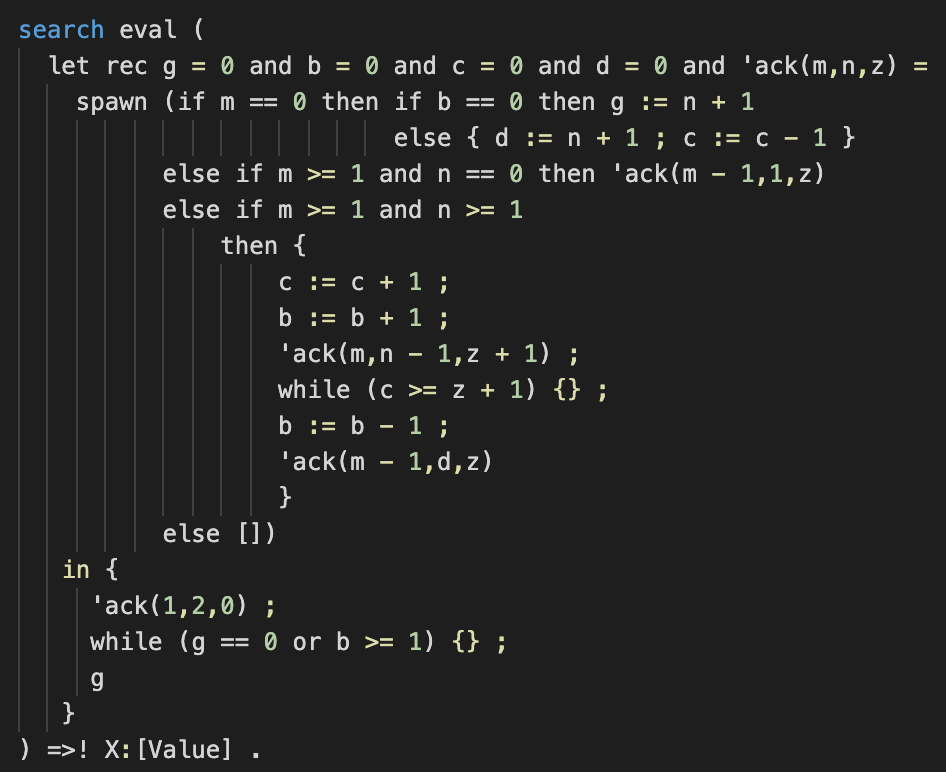
# Ejemplo Concurrencia:

Para la realización de este ejemplo se requiere crear un nuevo hilo de ejecución para cada llamada recursiva.

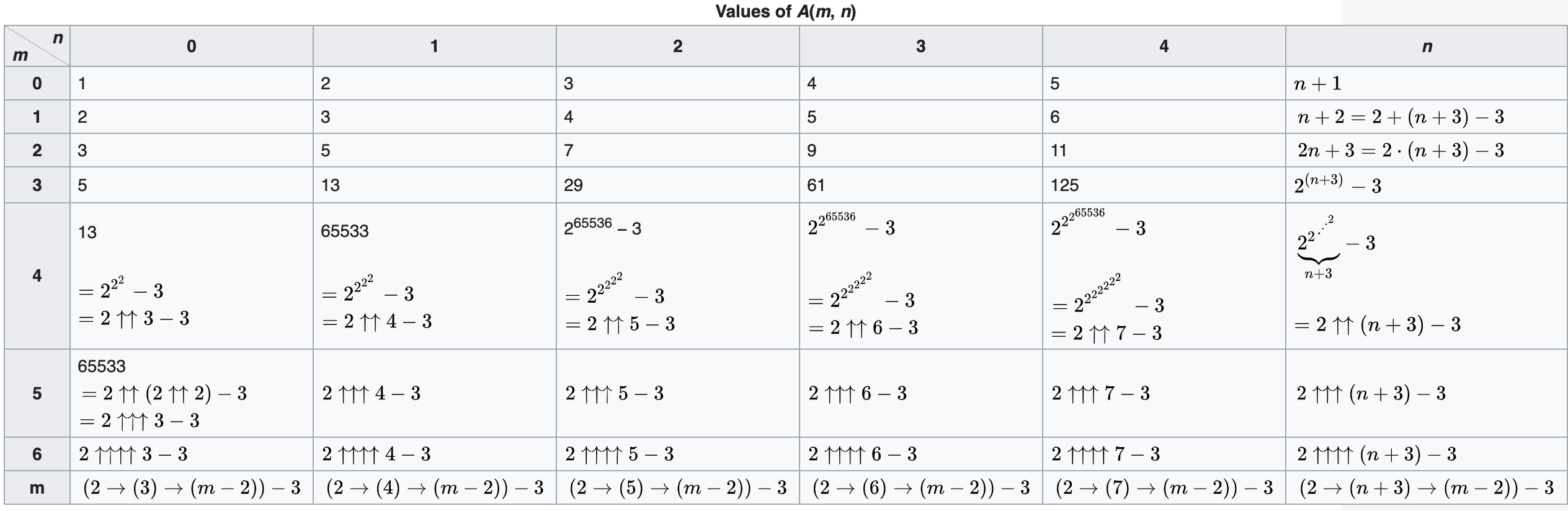
Por tanto, para evitar problemas con las variables, se ha modificado ligeramente el código anterior añadiendo una variable g que hará la función de la variable n pero de manera global, para que los distintos hilos sobrescriban g y no se pierdan los valores al hacer uso de la concurrencia.

Además de eso, para evitar condiciones de carrera y evitar que se obtengan resultados anómalos en las variables, se ha optado por la utilización de un bucle while, cuyo fin es el de solucionar estas situaciones.

De manera que el fragmento de código sería el siguiente:

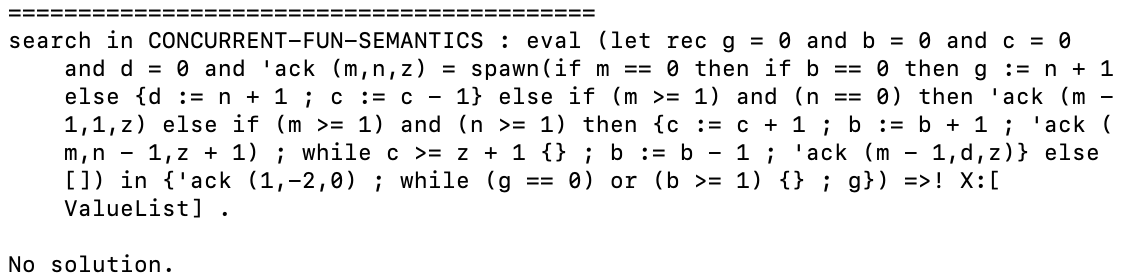


# Resultados obtenidos:

Para la correcta comprobación de los resultados se ha optado por utilizar la siguiente tabla para comprobar que los resultados obtenidos son correctos

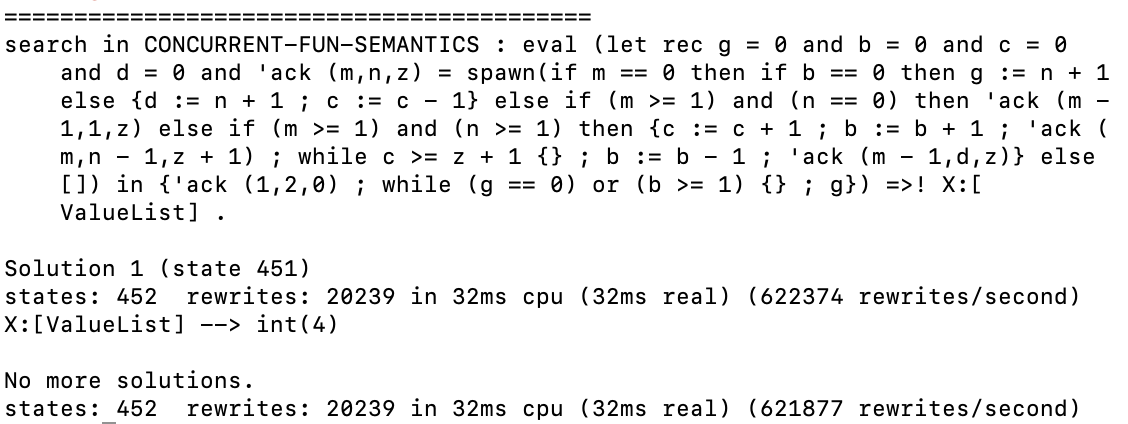
## Ejemplo 1:

Para esta ejecución se ha optado por la utilización de n= -2 y m = 1, de manera que al tratarse de una n con un valor negativo la ejecución no debe devolver ningún valor:



## Ejemplo 2:

Para esta ejecución se ha optado por la utilización de n= 2 y m = 1, de manera que al tratarse de una n con un valor positivo y comprobando la tabla anterior la ejecución debería devolver el valor 4.



## Ejemplo 3:

Para esta ejecución se ha optado por la utilización de n = 1 y m = 2, de manera que al tratarse de una n con un valor positivo y comprobando la tabla anterior la ejecución debería devolver el valor 5.

Imagen que contiene texto

Descripción generada automáticamente

## Ejemplo 4:

Para esta ejecución se ha optado por la utilización de n = 2 y m = 2, de manera que al tratarse de una n con un valor positivo y comprobando la tabla anterior la ejecución debería devolver el valor 7.

Imagen que contiene texto

Descripción generada automáticamente