Creación de un AFN a partir de una expresión regular: El AFN tiene como máximo el doble de estados que el número de símbolos y operadores de la expresión regular. El AFN tiene un estado inicial y un estado final. tiene tan solo una transición con un símbolo del alfabeto, o dos transiciones λ

Creación de AFD a partir de un AFN: General el estado inicial del AFD como el macroestado inicial del AFN incluirlo en una lista de estados por analizar. Analizar el primer estado de la lista por analizar: extraer el primer estado de la lista e introducirlo en la lista de estados analizados. Estudiar las transiciones del Estado para cada símbolo del alfabeto. Si el macroestado correspondiente a una transición no ha aparecido con anterioridad, crear un nuevo estado del AFD correspondiente a dicho macroestado e incluirlo en la lista de estados por analizar.Repetir el paso anterior hasta que no queden estados por analizar. Los estados finales del AFD son aquellos que corresponden a macroestados que contengan algún estado final del AFN.

Creación de AFD a partir de una expresion regular: Se introduce un punto en la expresión regular para indicar la parte reconocida en cada momento.Estado del autómata está asociado a un conjunto de expresiones regulares con puntos. El estado inicial se obtiene colocando el punto al comienzo de la expresión regular. Las transiciones de cada estado corresponden al consumo de algún símbolo, y dan lugar al desplazamiento del punto en la expresión regular.

Minimizacion de estados de un AFD: 1 - crear una partición en dos grupos: estados finales y no finales. 2 - Para cada grupo con varios estados, dividir el grupo en subgrupos tales qué dos estados, S&T, estén en el mismo subgrupo sí y sólo si para cada símbolo A, si existe la transición desde S con el símbolo Ahacia un estado de un cierto grupo, entonces debe existir la transición desde T con el símbolo A hacia un estado del mismo grupo. 3 - Repetir el paso 2 hasta que no se dividan en más grupos. 4 - Cada grupo representa un estado del AFD minimizado. 5 - Eliminar los estados no alcanzables desde el estado inicial y los que no tengan transiciones que puedan conducir a un estado final.

Memoria de pila: Se introdujo para manejar lenguajes estructurados con llamadas recursivas. Las funciones recursivas requieren manejar diferentes instancias del registro de activación, es decir, del conjunto de valores de sus variables en cada ejecución de la función. Estos registros de activación se almacenan en forma de pila, de manera que el registro de la cima de la pila corresponde a la función en ejecución. Al terminar la ejecución de una función debe desapilarse el registro de activación y pasar el control a la función que se encuentre en la cima de la pila. Al traspasar el control de una función a otra es necesario almacenar el estado de la máquina, es decir, el conjunto de valores de los registros del procesador. También es necesario almacenar la dirección del comienzo del registro de activación de la función llamante (Frame Pointer), para poder retomar el control una vez terminada la función llamada. Esta información se añade al contenido del registro de activación de cada función.

Estructura registro activación: Parte colocada por la función llamada {Var temp, var loc, estado maqu} Parte colocada funcion que llama {Dir. ret, values param, value ret}

Llamada: – Se reserva espacio para el valor devuelto. – Se almacena el valor de los parámetros que se pasan a la función llamada. – Se almacena el valor de la dirección del código de la función que llama (dirección de retorno). – Se almacena el estado de la máquina (incluye SP y FP). – Se pasa el control a la función llamada. – Comienza la ejecución del código de la función llamada.

Retorno: – Se asigna el valor devuelto. – Se restaura el contenido del estado de la máquina. Esto modifica el valor de FP y SP y provoca la liberación de la memoria ocupada por el registro de activación de la función llamada. – Se restaura el valor de la dirección del código de la función que llama (dirección de retorno). – Se devuelve el control a la función que llama. – Se almacena el valor devuelto en la variable local o temporal adecuada

Creación de un AFN a partir de una expresión regular: El AFN tiene como máximo el doble de estados que el número de símbolos y operadores de la expresión regular. El AFN tiene un estado inicial y un estado final. tiene tan solo una transición con un símbolo del alfabeto, o dos transiciones λ

Creación de AFD a partir de un AFN: General el estado inicial del AFD como el macroestado inicial del AFN incluirlo en una lista de estados por analizar. Analizar el primer estado de la lista por analizar: extraer el primer estado de la lista e introducirlo en la lista de estados analizados. Estudiar las transiciones del Estado para cada símbolo del alfabeto. Si el macroestado correspondiente a una transición no ha aparecido con anterioridad, crear un nuevo estado del AFD correspondiente a dicho macroestado e incluirlo en la lista de estados por analizar.Repetir el paso anterior hasta que no queden estados por analizar. Los estados finales del AFD son aquellos que corresponden a macroestados que contengan algún estado final del AFN.

Creación de AFD a partir de una expresion regular: Se introduce un punto en la expresión regular para indicar la parte reconocida en cada momento.Estado del autómata está asociado a un conjunto de expresiones regulares con puntos. El estado inicial se obtiene colocando el punto al comienzo de la expresión regular. Las transiciones de cada estado corresponden al consumo de algún símbolo, y dan lugar al desplazamiento del punto en la expresión regular.

Minimizacion de estados de un AFD: 1 - crear una partición en dos grupos: estados finales y no finales. 2 - Para cada grupo con varios estados, dividir el grupo en subgrupos tales qué dos estados, S&T, estén en el mismo subgrupo sí y sólo si para cada símbolo A, si existe la transición desde S con el símbolo Ahacia un estado de un cierto grupo, entonces debe existir la transición desde T con el símbolo A hacia un estado del mismo grupo. 3 - Repetir el paso 2 hasta que no se dividan en más grupos. 4 - Cada grupo representa un estado del AFD minimizado. 5 - Eliminar los estados no alcanzables desde el estado inicial y los que no tengan transiciones que puedan conducir a un estado final.

Memoria de pila: Se introdujo para manejar lenguajes estructurados con llamadas recursivas. Las funciones recursivas requieren manejar diferentes instancias del registro de activación, es decir, del conjunto de valores de sus variables en cada ejecución de la función. Estos registros de activación se almacenan en forma de pila, de manera que el registro de la cima de la pila corresponde a la función en ejecución. Al terminar la ejecución de una función debe desapilarse el registro de activación y pasar el control a la función que se encuentre en la cima de la pila. Al traspasar el control de una función a otra es necesario almacenar el estado de la máquina, es decir, el conjunto de valores de los registros del procesador. También es necesario almacenar la dirección del comienzo del registro de activación de la función llamante (Frame Pointer), para poder retomar el control una vez terminada la función llamada. Esta información se añade al contenido del registro de activación de cada función.

Estructura registro activación: Parte colocada por la función llamada {Var temp, var loc, estado maqu} Parte colocada funcion que llama {Dir. ret, values param, value ret}

Llamada: – Se reserva espacio para el valor devuelto. – Se almacena el valor de los parámetros que se pasan a la función llamada. – Se almacena el valor de la dirección del código de la función que llama (dirección de retorno). – Se almacena el estado de la máquina (incluye SP y FP). – Se pasa el control a la función llamada. – Comienza la ejecución del código de la función llamada.

Retorno: – Se asigna el valor devuelto. – Se restaura el contenido del estado de la máquina. Esto modifica el valor de FP y SP y provoca la liberación de la memoria ocupada por el registro de activación de la función llamada. – Se restaura el valor de la dirección del código de la función que llama (dirección de retorno). – Se devuelve el control a la función que llama. – Se almacena el valor devuelto en la variable local o temporal adecuada