Memoria p1.

Conversión AFND - AFD

ANTONIO

CRISTINA

1. **Algoritmo**

Teniendo un AFND, formamos una lista con los estados que tiene, cada estado se compone de un nombre,

tipo y otra lista con las transiciones, siendo el resultado una “función de transición”. Empezamos a extraer

para cada estado de la lista los datos para analizarlos e insertarlos en otra nueva lista que corresponderá a

la función de transición del AFD.

Teniendo que:

Si el estado de la lista tiene como lambda uno de los siguientes, se fusionan como un mismo

estado.

Si encontramos una transición distinta al estado de la lista se genera un nuevo estado con esa transición y

se añade a la nueva lista.

La nueva lista compone el AFD, por tanto, extraemos los datos de la lista generando el nuevo autómata

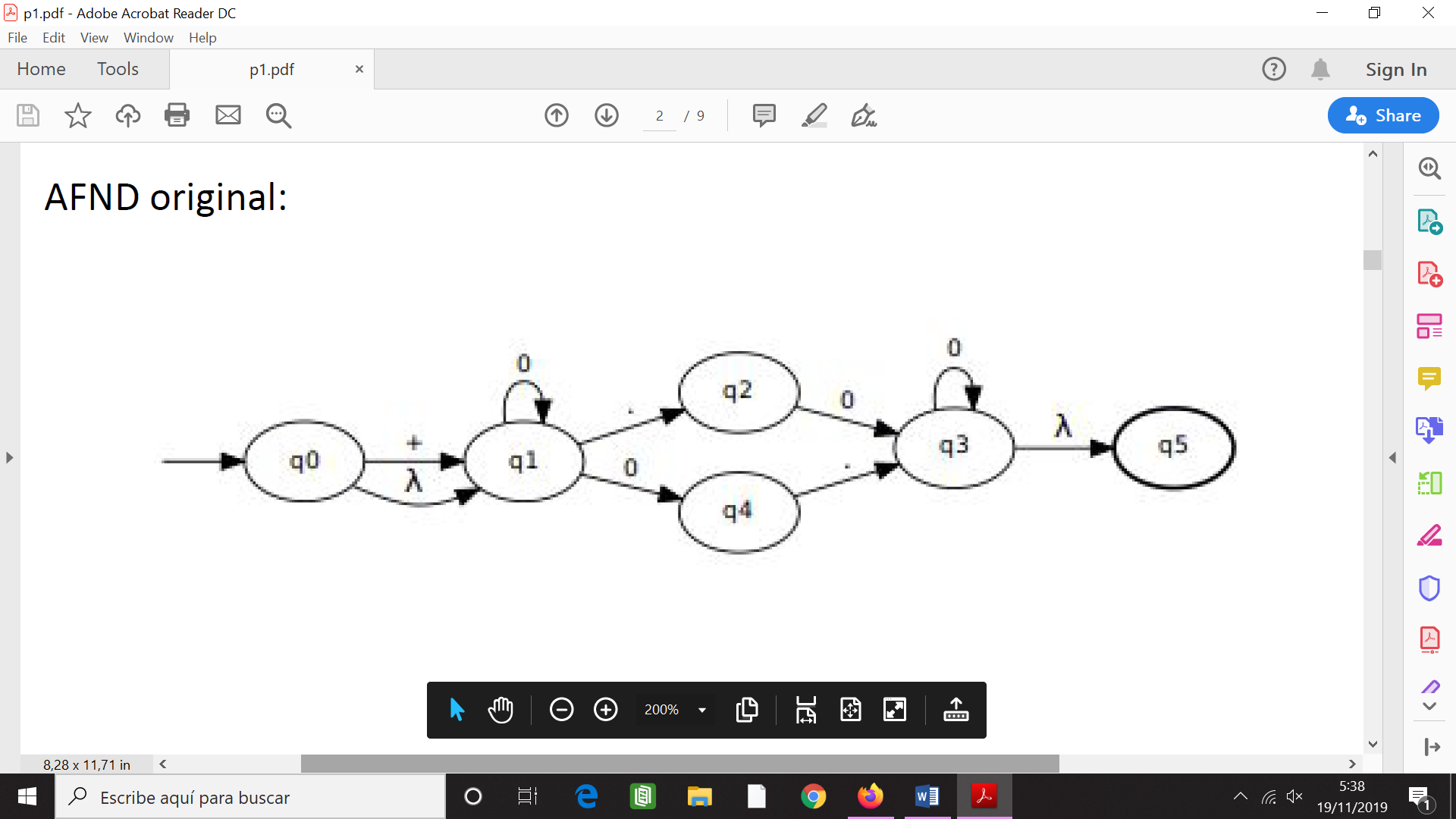
mediante las funciones de afnd(.c/.h).

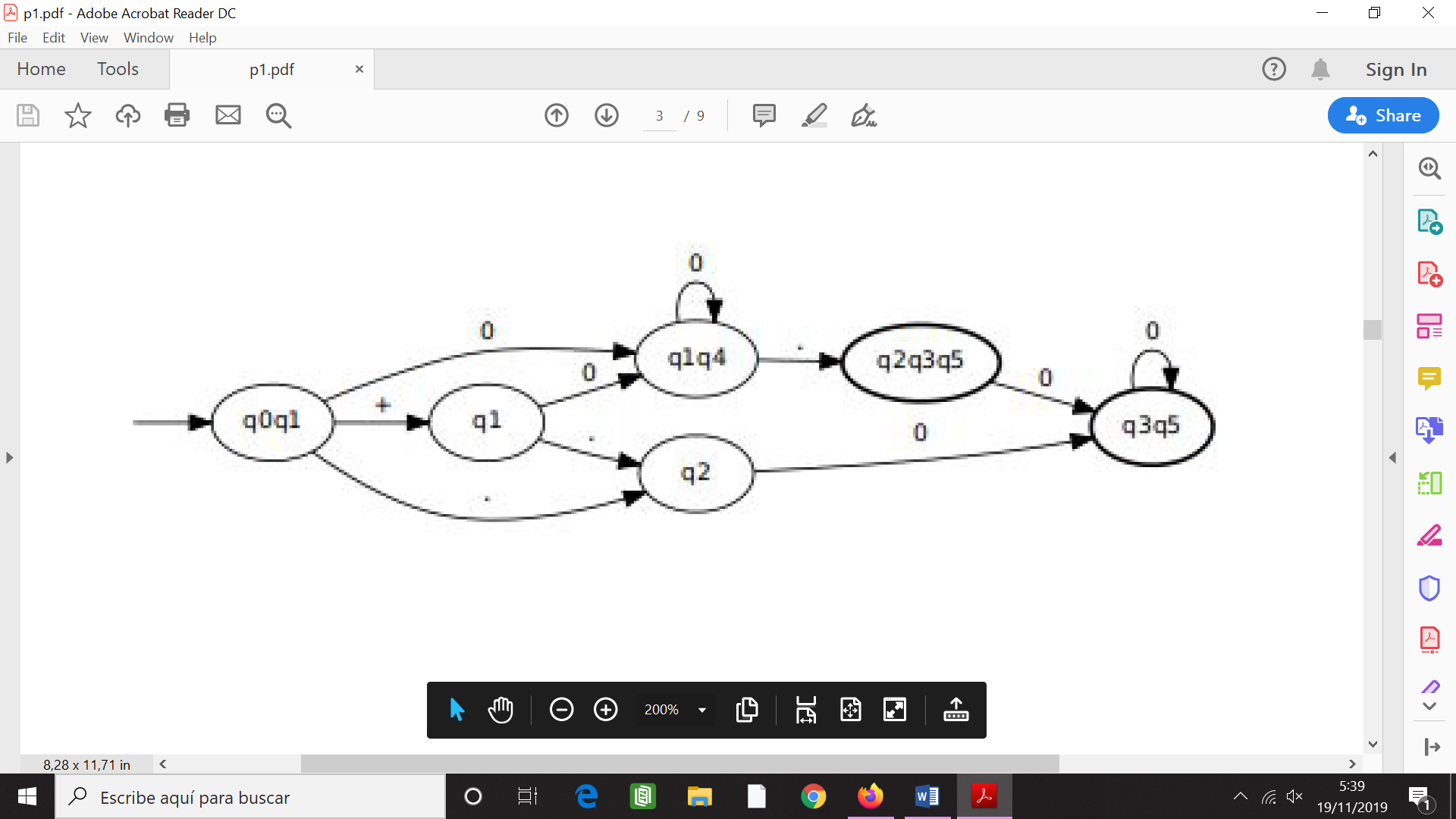
1. **Explicación funciones**

La implementación se compone de afnd(.c/.h), transforma(.c/.h), afnd\_plus(.c/.h), main.c y makefile.

* afnd(.c/.h) compone las funciones que implementan un AFND.
* transforma(.c/.h) compone:
* Una estructura intermedia con las características de un estado (nombre, tipo, lista de estados a los que transita, tamaño de la lista).
* void print\_state(const state \*s): Imprime los elementos de la estructura para un estado.
* Bool contains(state \* transitions, int transitions\_n, char \* state\_name): Comprueba si se encuentra el estado state\_name en la lista de transiciones.
* AFND \*AFNDTransforma(AFND \*afnd): Función que genera la conversión de AFND a AFD recurriendo a las funciones implementadas en afnd\_plus(.c/.h).
* afnd\_plus(.c/.h) compone:
* int get\_transitions(AFND \*afnd, int \*\*t\_list, int state, int symbol\_i): Genera una lista de transiciones para un estado.
* int get\_transitions\_x(AFND \*afnd, int \*\*t\_list, const int \*states, int num\_states, int symbol\_i): Genera una lista de transiciones para x estados.
* char \*gen\_name(AFND \*afnd, int \*states, int num\_states): Genera un nombre para un estado.
* int get\_states\_connected(AFND \*afnd, int \*\*states, int state): Genera una lista de transiciones mediante lambda.
* int gen\_type(AFND \*afnd, int \*states, int state\_n): Genera un tipo para un estado.
* int add\_lambdas(AFND \*afnd, int \*\*lstates, const int \*states, int states\_n): Dada una lista de estados, devuelve la lista más todas las conexiones lambda de cada estado.
* main.c se compone de varias funciones AFND(1…4) que son los 4 autómatas a probar.

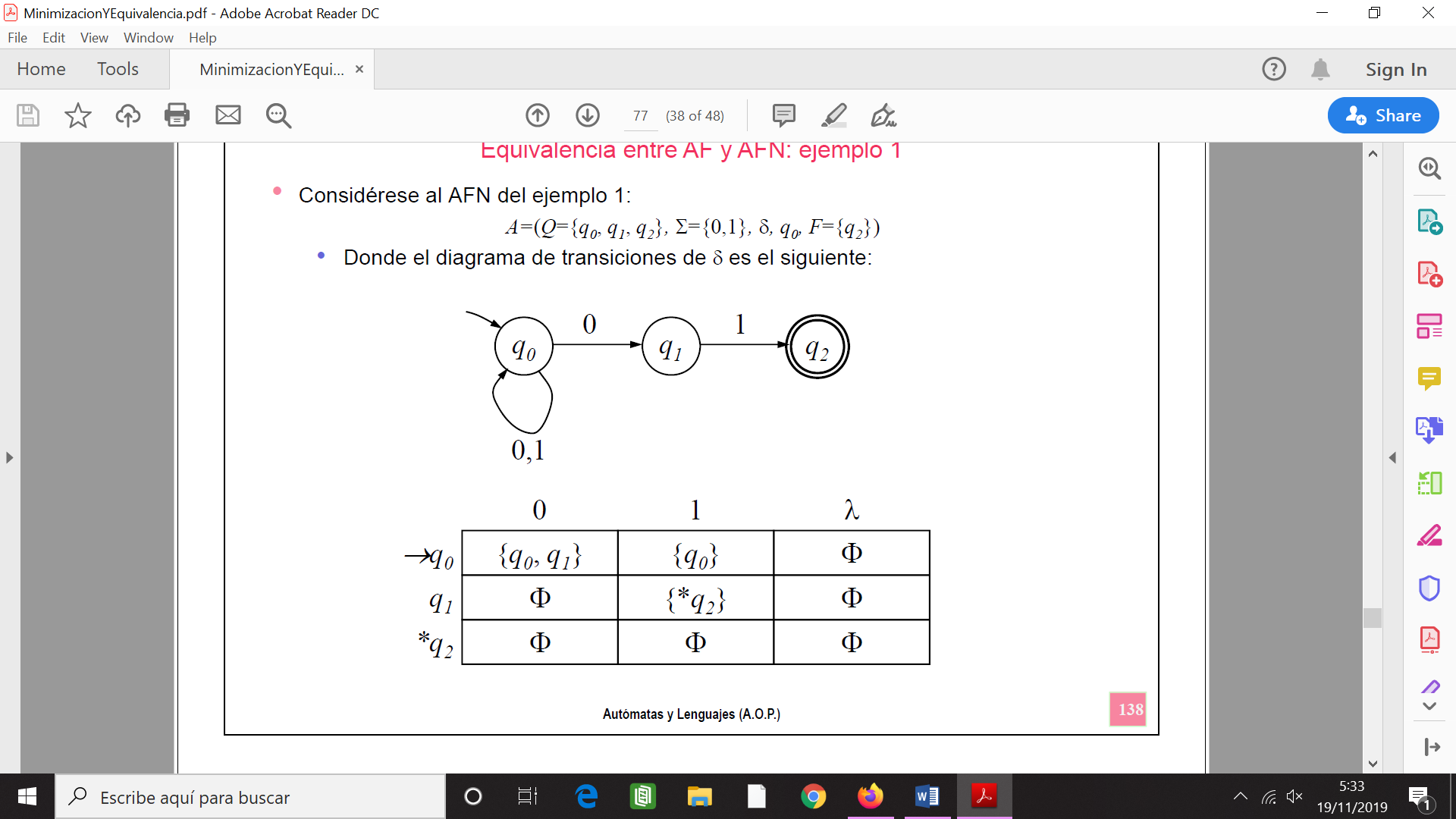
1. **Pruebas**
   1. Autómata 1.



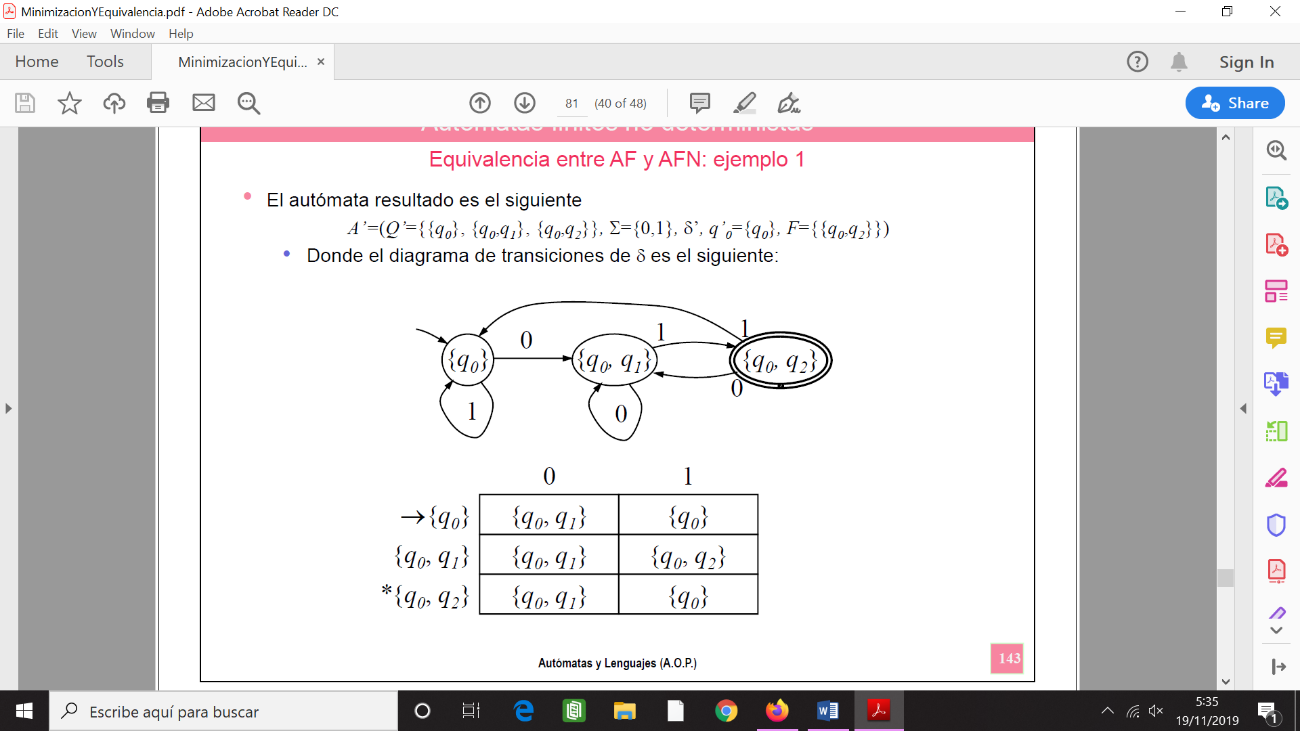


* 1. Autómata 2.

AFND:

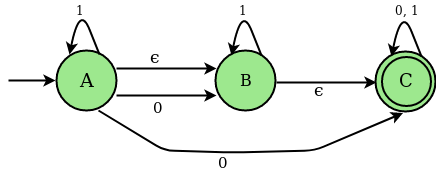


AFD:

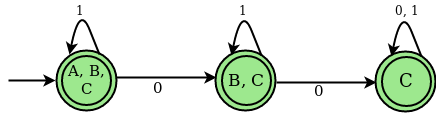


* 1. Autómata 3.

Ejemplo de: <https://www.geeksforgeeks.org/program-implement-nfa-epsilon-move-dfa-conversion/>

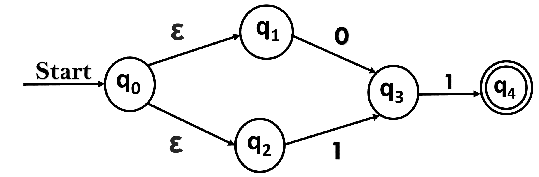
AFND:

AFD:



* 1. Autómata 4

Ejemplo de: <https://www.javatpoint.com/automata-conversion-from-nfa-with-null-to-dfa>

AFND:

AFD:

