--Parser de Gramáticas Regulares--

Para hacer el parser de Gramáticas Regulares se tuvieron en cuenta dos diseños:

-El primero era uno donde Lex tendría poca importancia, se lo utilizaría simplemente para levantar caracteres desde el stream de datos y luego mediante programación en C, decidir el estado del mismo.

-EL segundo era uno donde toda la lógica estaría en el código de Lex y sería ayudado por funciones de C.

Se optó por la segunda por ser mucho más simple la implementación, aunque al final fuera mucho más difícil de seguir el código. Aparte, de esta manera, se aprovecharía mejor el poder de Lex. Ya que de todas formas habría que usarlo y la cantidad de código autogenerado por Lex es considerable, era mejor hacer uso entero de esta circunstancia e implementar lo más posible dentro del marco que provee.

A su vez fue creado siguiendo el "camino feliz" de la ejecución. Eso quiere decir que Lex sólo parseará lo que esté correctamente escrito. Aquello que no lo esté, será trabajo del código en C identificar las razones de por qué no pudo ser levantado.

-Características-

El parser de gramáticas regulares es capaz de interpretar sólo gramáticas bien construidas. Las demás fallaran al pasar por el parser. Ante el más mínimo error, el programa abortará y informando en la mejor manera posible de la falla.

Debido a que el esqueleto de la lógica fue construido en Lex y por la forma de parseo, es imposible detectar la falla puntual en algunos casos. Sin embargo, es posible saber a grandes rasgos dónde se originó la falla, e informarle al usuario, a grandes rasgos, la razón principal del problema.

Esto se logra utilizando flags, que indiquen el último lugar que el programa logró levantar de forma correcta. Para casos más particulares, a los cuales Lex se le pueden escapar, como por ejemplo, el uso de símbolos no definidos previamente, o reglas inconsistentes, es que se utiliza el código en C para validar.