**Contenido:**

* Instruccion
* BackGround del problema
* Requerimientos:
* Brief description:
* Inputs:
* Outputs:

**Introduccion:**

**¿Que es un FBF?**

Todos los lenguajes se componen de unos símbolos y de unas reglas sintácticas que nos indican qué combinaciones de símbolos son correctas y cuáles no lo son. Por ejemplo, en [castellano](http://www.monografias.com/trabajos5/oriespa/oriespa.shtml) no podemos decir:

Mis amigos y yo voy al cine

La oración del ejemplo está mal formada porque no hay la concordancia debida entre el número del sujeto (plural) y el número del verbo (singular). También en matemáticas hay unas reglas que nos indican qué combinaciones de símbolos podemos hacer, de modo que si nos presentarán lo siguiente:

%=4+(78-)

no sabríamos qué hacer simplemente porque la expresión está mal formada, no respeta las reglas de formación de fórmulas matemáticas. Del mismo modo, cualquier combinación de símbolos lógicos no constituye una fórmula bien formada. Así por ejemplo, no están bien formadas las fórmulas:

Ap

vpvq

p-> ¬

etc…

No es difícil descubrir intuitivamente, a partir de ejemplos, qué fórmulas están bien formadas en lógicas y cuáles no, pero no está de más ofrecer las siguientes **reglas para la formación de fórmulas bien formadas** (fbf):

Regla 1: Toda proposición atómica es una fbf

Regla 2: Si A es una fbf, entonces ¬A también es una fbf

Regla 3: Si A y B son fbf, entonces (AvB), (A->B) y (A^B) también son fbf

Leer más: <http://www.monografias.com/trabajos-pdf5/la-logica-proposicional/la-logica-proposicional.shtml#ixzz3WeHZTryE>

**BackGround del problema:**

una persona para poder identificar si una proposición es tautológica y o contradictoria, primero tiene que identificar si la misma es una fórmula bien formada (FBF). ¿que pasaria si la preposición a verificar es muy larga?, la persona estaría dispensa a errores, errores humanos. recibir un listado de proposiciones sería algo peor ya que también nosotros los humanos nos cansamos y tendemos a dejadez; visto esto se nos genera una pregunta: ¿Acaso no sería de utilidad una herramienta que nos permita identificar primero si la preposiciones estan bien formadas?

**Requerimientos**:

1. Debe comprobar si uno o más proposiciones es un FBF
2. Si una preposicion es un FBF, debe de especificar el porqué; en caso de no se un FBF también debe de especificarlo.
3. A Partir de algún archivo de texto tomar un listado y crear otro con los resultados obtenidos.
4. El programa debe de estar listo para maximo 2 semanas.
5. EL programa no tendrá interfaz visual, en caso de no ser así entonces que sea en consola.
6. El usuario se limita a brindarnos los nombre del archivo de entrada y el archivo de salida.
7. El programa debe ser modificable para futuras agregación de reglas a esta.

**Brief description:**

El programa debe de tomar un archivo de texto (input) donde este archivo contendrá un un listado de proposiciones a evaluar. Luego de evaluar cada una de las proposiciones, este crea una archivo de texto(output) con los resultados obtenidos, especificando si la preposicion evaluada es un FBF, describiendo paso a paso cada símbolos lógicos (p,q,f,r etc) y sus relaciones se componen alguna propiedad de FBF.

***Por ejemplo:***

***En el interior del archivo “Input.txt” se encuentra:***

~p(p!q)

(~p->(pvq)

q-> (~p->(pvq))

***En el interior del archivo “output.txt” se encuentra:***

~p(p!q) ----> No es FBF

no hay un conector entre ~p y (p!q).

la relación ! entre p y q no existe.

q-> (~p->(pvq) ----> No es FBF

los paréntesis no estan balanceados

q-> (~p->(pvq)) ----> Es un FBF

p ----> Es un FBF

q ----> Es un FBF

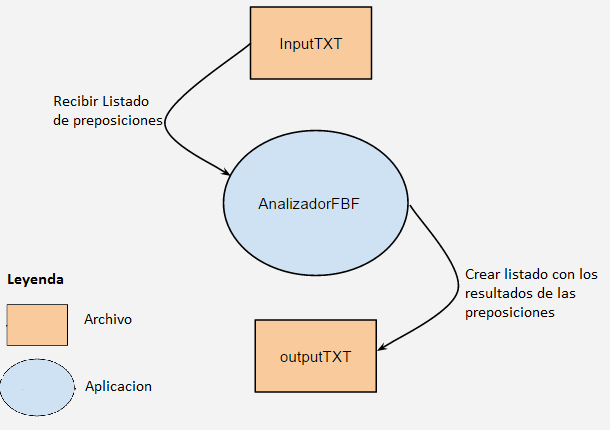
~p ----> Es un FBF

(pvq) ----> preposicion atómica, es un FBF

(~p->(pvq)) ----> preposicion atómica, es un FBF

*Context diagram:*

Este diagrama es para que se puede percibir más el entorno de cómo sea el flujo de la aplicacion.



**Inputs:**

La entrada del programa será un archivo .txt con un listado de proposiciones separados por una nueva línea. En caso de que este archivo no exista entonces no se pude proceder con el análisis del mismo.

**Sintaxis de las preposiciones:**

cualquier letra, palabra o frase es aceptado como variable, sea mayuscula o minuscula (q, p, r, mujer, hombre, carlito, etc)

*Ejemplo:*

p->q

(Juan ^ pedro)->pájaros

Conectores y negación:

v = disyunción “ó”

^ = conjunción “y”

-> = condicional “si entonces”

<-> = bicondicional “si entonces si”

~ = Negacion

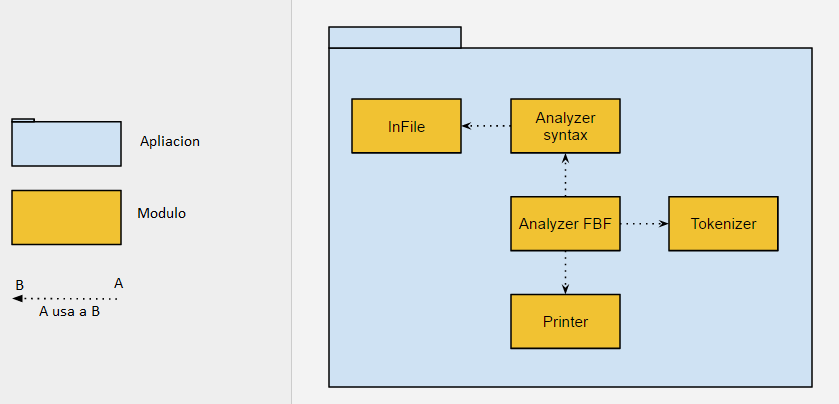
Para la agrupación de las proposiciones atómicas estarán representadas por paréntesis().

**Outputs:**

El resultado del archivo analizado es otro archivo .txt con los resultados impresos de cada variable individual y agrupación conformado por la variable y su conector (preposicion atómica).

Para más información ir al ejemplo del Brief description.

**Module view Use Style:**



**Descriptions:**

Estos modulos representan las distintas clases de la aplicacion, el objetivo aqui es mantener la modificabilidad y la portabilidad, y sobre todo identificar las dependencia de cada módulo hacia otro.

**InFile:**

Se encarga de la captura completa de las proposiciones en el archivo. A partir de las proposiciones escritas en el archivo esta crea una lista con las proposiciones para su manejamiento.

**AnalyzerSintax:**

Se encarga de a analizar la sintaxis de las prepocisiones. verifica si hay incoherencias con respecto a la sintaxis, las relaciones de las variables en conjunto con los conectores y los paréntesis.

**Tokenizer:**

Identifica los token que conforman la preposicion (símbolos y id) y crea una lista con ellas.

**AnalizerFBF:**

Se encarga de analizar cada variable, las relación entre las mismas(proposición atómica). identifica en la proposición las propiedades que la conforman. Justifica el porque es un FBF.

**Printer:**

Crea un archivo nuevo de texto con las justificaciones de que si es FBF o no. Recibe las propiedades identificadas por el AnalyzerFBF para ser impresa en el archivo nuevo.