



Licenciatura en Sistemas

Sistemas basados en conocimiento

Trabajo Final

“Test de Inconsistencias”

Docente

Hernán Amatriain

Alumnos

Guido Ezequiel Contento

Luciano Otegui

Introducción

El presente documento detalla la composición y el funcionamiento del “Test de Inconsistencias” desarrollado para la detección de inconsistencias en reglas de un sistema basado en conocimientos. Se presenta a continuación la estructura del proyecto, los métodos principales del mismo, y los pasos para probar el mismo.

Estructura del proyecto

El proyecto se trata de una aplicación de consola desarrollada en el lenguaje C# utilizando el framework Net Core 5.0. Al abrir la solución del mismo, encontraremos las siguientes carpetas:

- Constants: Posee la clase **Constants.cs** donde se guardan la ruta de lectura del archivo de entrada, y las constantes necesarias para extraer las reglas de este archivo.
- Entities: Posee las clases **Rule.cs** (representa una regla extraída del archivo) y **EqualComponentRule.cs** (representa dos reglas emparejadas que tienen un mismo antecedente o consecuente), utilizadas durante la ejecución del programa.
- Helpers: Posee las clases **FileHelper.cs** (encargada de extraer las reglas del archivo de entrada) y **FunctionHelper.cs** (clase principal que contiene los métodos para detección de inconsistencias).

La ejecución del programa se hace desde el método Main de la clase **Program.cs**.

Métodos principales

Los métodos principales que utiliza el sistema se hayan en la clase FunctionHelper.cs. Cabe destacar que el sistema solamente puede detectar 4 tipos de inconsistencias: Reglas redundantes, Reglas incluidas en otras, Reglas conflictivas y Condiciones Si innecesarias. Los métodos principales se detallan a continuación

- ***public static bool AntecedentsOrConsequentsAreEqual(Rule r1, Rule r2, bool isAntecedentCheck);***

Este método recibe dos reglas, y un booleano que indica si se debe chequear el antecedente o el consecuente de las reglas. Se encarga de revisar si los elementos que componen los antecedentes o consecuentes a ambas reglas, sin importar su orden, son iguales o no.

- ***public static void FindReglasRedundantes(List<EqualComponentRule> pairedRules);***

Este método recibe un listado de reglas emparejadas, donde cada par de reglas posee el mismo antecedente. Se encargará de determinar si en cada par suministrado, las reglas son redundantes (es decir, si una o más postcondiciones de su consecuente son iguales). También se alerta en caso de que se trate de la misma regla. Si no hay condición de redundante, o las reglas no son la misma, no se devuelve ninguna información.

- ***public static void FindReglasConflictivas (List<EqualComponentRule> pairedRules);***

Este método recibe un listado de reglas emparejadas, donde cada par de reglas posee el mismo antecedente. Se encargará de determinar si para cada par de reglas, los consecuentes presentan condiciones que determinen que esas dos reglas son conflictivas (es decir, si algún elemento los consecuentes de las reglas se contradicen). También se alerta en caso de que las reglas sean iguales. Si no hay condición de conflictiva, entonces no se devuelve ninguna información.

- ***public static void FindReglasIncluidasEnOtras (List<EqualComponentRule> pairedRules);***

Este método recibe un listado de reglas emparejadas, donde cada par de reglas posee el mismo consecuente. Se encargará de determinar si para cada par de reglas, los antecedentes presentan condiciones que las conviertan en reglas incluidas en otras (es decir, si todo el antecedente de una regla se encuentra dentro de otra regla). También se alerta en caso de que las dos reglas comparadas sean iguales. En caso de no haber condición de regla incluida en otra, o si las reglas no son la misma, no se devuelve información.

- ***public static void FindCondicionesSiInnecesarias (List<EqualComponentRule> pairedRules);***

Este método recibe un listado de reglas emparejadas, donde cada par de reglas posee el mismo consecuente. Se encargará de determinar si para cada par de reglas, los antecedentes presentan condiciones que las conviertan en condiciones si innecesarias (es decir, si algún elemento del antecedente se encuentra negado en las reglas, y si al eliminar estos, el resto de las precondiciones son iguales). También se alerta en caso de que las dos reglas comparadas sean iguales. En caso de no haber condición si innecesaria, o si las reglas no son la misma, no se devuelve información.

Prueba del sistema

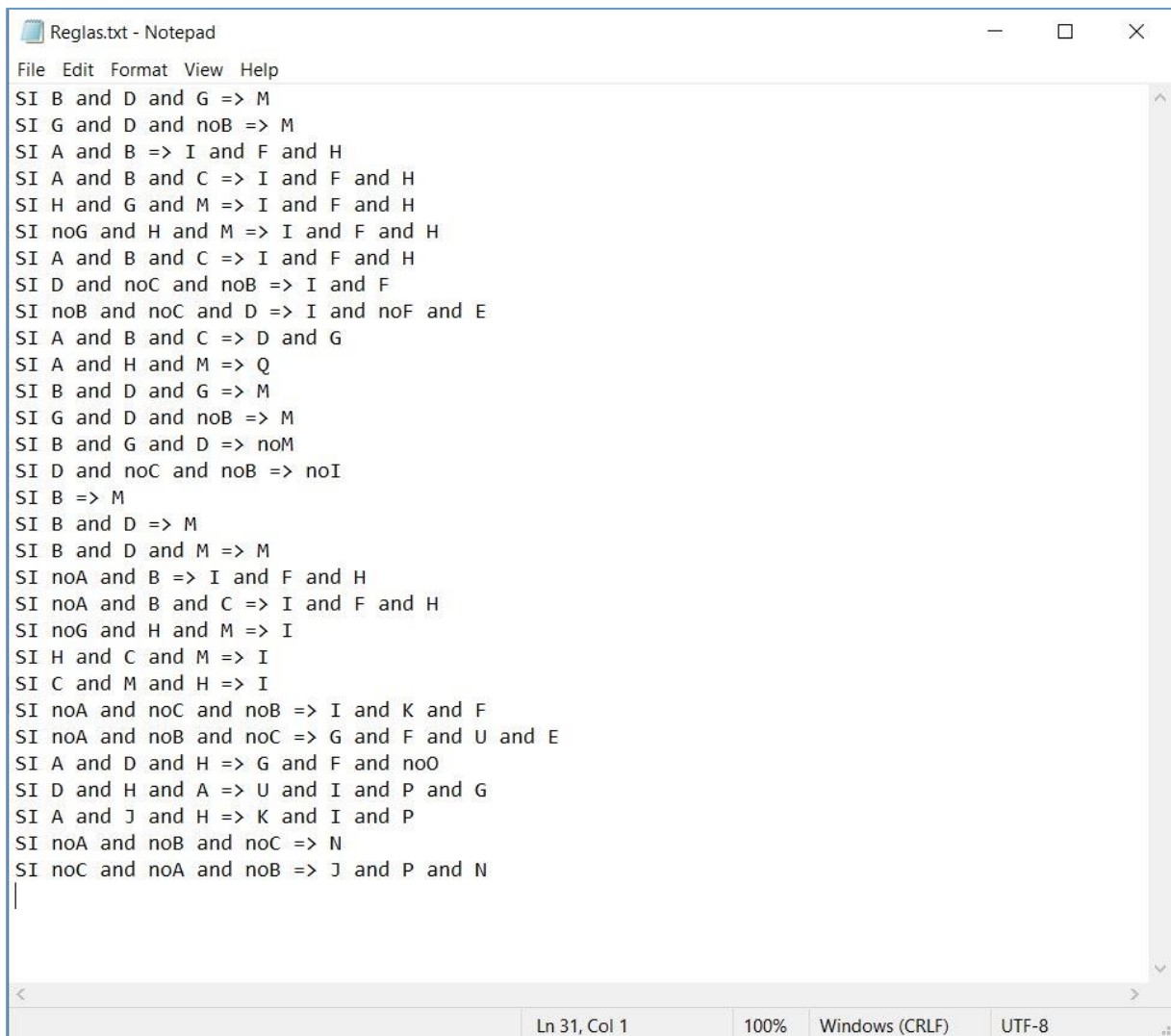
Para realizar una prueba del sistema, se debe tener un archivo de texto que posea reglas en el siguiente formato:

Si <ANTECEDENTE> => <CONSECUENTE>

Por ejemplo:

Si A and G and noH => F and D

La carpeta “Reglas y Documentos” incluida en el proyecto, presenta cinco archivos, donde en cuatro de ellos hay reglas que poseen diferentes inconsistencias, según el nombre del archivo, de forma que se puede probar un archivo concreto en caso de que se solo se desee buscar un tipo de inconsistencia. El archivo “Reglas.txt” se trata de una unión de los otros cuatro archivos. El archivo de entrada debe listar las reglas de la siguiente manera:



```
File Edit Format View Help
SI B and D and G => M
SI G and D and noB => M
SI A and B => I and F and H
SI A and B and C => I and F and H
SI H and G and M => I and F and H
SI noG and H and M => I and F and H
SI A and B and C => I and F and H
SI D and noC and noB => I and F
SI noB and noC and D => I and noF and E
SI A and B and C => D and G
SI A and H and M => Q
SI B and D and G => M
SI G and D and noB => M
SI B and G and D => noM
SI D and noC and noB => noI
SI B => M
SI B and D => M
SI B and D and M => M
SI noA and B => I and F and H
SI noA and B and C => I and F and H
SI noG and H and M => I
SI H and C and M => I
SI C and M and H => I
SI noA and noC and noB => I and K and F
SI noA and noB and noC => G and F and U and E
SI A and D and H => G and F and noO
SI D and H and A => U and I and P and G
SI A and J and H => K and I and P
SI noA and noB and noC => N
SI noC and noA and noB => J and P and N
|
```

Inicialmente, debemos definir la Ruta que se utilizará para leer el archivo, ya que este se analiza al ejecutar el programa. Debemos setearla en la clase **Constants.cs**:

```
public static class Constants
{
    public static readonly string FILE_ROUTE = "C:\\Proyectos\\UNLa\\SBEC\\Reglas y Documentos\\Reglas.txt";
}
```

En este caso, seteamos el archivo "Reglas.txt" que contiene todos los tipos de inconsistencias. A continuación, ejecutamos la aplicación, y si el formato del archivo está correcto, las reglas leídas se imprimirán por pantalla:

```

Leyendo archivo de reglas...
Archivo leído con éxito
Reglas:
Regla 1: B and D and G => M
Regla 2: G and D and noB => M
Regla 3: A and B => I and F and H
Regla 4: A and B and C => I and F and H
Regla 5: H and G and M => I and F and H
Regla 6: noG and H and M => I and F and H
Regla 7: A and B and C => I and F and H
Regla 8: D and noC and noB => I and F
Regla 9: noB and noC and D => I and noF and E
Regla 10: A and B and C => D and G
Regla 11: A and H and M => Q
Regla 12: B and D and G => M
Regla 13: G and D and noB => M
Regla 14: B and G and D => noM
Regla 15: D and noC and noB => noI
Regla 16: B => M
Regla 17: B and D => M
Regla 18: B and D and M => M
Regla 19: noA and B => I and F and H
Regla 20: noA and B and C => I and F and H
Regla 21: noG and H and M => I
Regla 22: H and M and C => I
Regla 23: H and M and C => I
Regla 24: noA and noC and noB => I and K and F
Regla 25: noA and noB and noC => G and F and U and E
Regla 26: A and D and H => G and F and noO
Regla 27: D and H and A => U and I and P and G
Regla 28: A and J and H => K and I and P
Regla 29: noA and noB and noC => N
Regla 30: noC and noA and noB => J and P and N
=====

```

Debajo de las reglas, encontramos un menú que nos permitirá seleccionar que tipo de Inconsistencia queremos detectar. En este caso vamos a buscar la presencia de **Reglas Redundantes**, por lo que escribimos 1 y presionamos Enter:

```

=====Detector de Inconsistencias=====
Elija opcion:
  1) Reglas Redundantes
  2) Reglas Conflictivas
  3) Reglas Incluidas En Otras
  4) Condiciones SI Innecesarias
  0) Salir
==> 1

```

Finalmente, se muestra los resultados del análisis:

```
==Chequeando Reglas Redundantes==  
  
Regla -1- Y Regla -12- tienen el mismo consecuente por lo que las reglas son iguales.  
Regla -2- Y Regla -13- tienen el mismo consecuente por lo que las reglas son iguales.  
Regla -4- Y Regla -7- tienen el mismo consecuente por lo que las reglas son iguales.  
I de Regla -6- encontrada en Regla -21- . SON REDUNDANTES  
I de Regla -8- encontrada en Regla -9- . SON REDUNDANTES  
Regla -22- Y Regla -23- tienen el mismo consecuente por lo que las reglas son iguales.  
F de Regla -24- encontrada en Regla -25- . SON REDUNDANTES  
G de Regla -26- encontrada en Regla -27- . SON REDUNDANTES  
N de Regla -29- encontrada en Regla -30- . SON REDUNDANTES  
  
==Fin del análisis de reglas redundantes==
```

Análisis de los resultados

Regla 1: $B \text{ and } D \text{ and } G \Rightarrow M$

Regla 12: $B \text{ and } D \text{ and } G \Rightarrow M$

Las reglas son iguales.

Regla 2: $G \text{ and } D \text{ and } \text{no}B \Rightarrow M$

Regla 13: $G \text{ and } D \text{ and } \text{no}B \Rightarrow M$

Las reglas son iguales.

Regla 4: $A \text{ and } B \text{ and } C \Rightarrow I \text{ and } F \text{ and } H$

Regla 7: $A \text{ and } B \text{ and } C \Rightarrow I \text{ and } F \text{ and } H$

Las reglas son iguales.

Regla 6: $\text{no}G \text{ and } H \text{ and } M \Rightarrow I \text{ and } F \text{ and } H$

Regla 21: $\text{no}G \text{ and } H \text{ and } M \Rightarrow I$

El componente I está en ambas reglas, por lo que son redundantes.

Regla 8: $D \text{ and } \text{no}C \text{ and } \text{no}B \Rightarrow I \text{ and } F$

Regla 9: $\text{no}B \text{ and } \text{no}C \text{ and } D \Rightarrow I \text{ and } \text{no}F \text{ and } E$

El componente I está en ambas reglas, por lo que son redundantes.

Regla 24: $\text{no}A \text{ and } \text{no}C \text{ and } \text{no}B \Rightarrow I \text{ and } K \text{ and } F$

Regla 25: $\text{no}A \text{ and } \text{no}B \text{ and } \text{no}C \Rightarrow G \text{ and } F \text{ and } U \text{ and } E$

El componente F está en ambas reglas por lo que son redundantes.

Regla 26: $A \text{ and } D \text{ and } H \Rightarrow G \text{ and } F \text{ and } \text{no}O$

Regla 27: $D \text{ and } H \text{ and } A \Rightarrow U \text{ and } I \text{ and } P \text{ and } G$

El componente G está en ambas reglas, por lo que son redundantes.

Regla 29: $\text{no}A \text{ and } \text{no}B \text{ and } \text{no}C \Rightarrow N$

Regla 30: $\text{no}C \text{ and } \text{no}A \text{ and } \text{no}B \Rightarrow J \text{ and } P \text{ and } N$

El componente N está en ambas reglas, por lo que son redundantes.

Comparando el resultado obtenido en la prueba, con el análisis posterior realizado, podemos afirmar que tanto las redundancias como los pares de reglas iguales detectados por el sistema, son correctos.

Código del Proyecto

El repositorio con el código del proyecto se encuentra disponible en:
<https://github.com/Luchoooo98/SBEC>