Universidad de Buenos Aires

Facultad de Ingeniería



75.69 - Sistemas Automáticos de Diagnóstico y Detección de Fallas II

Trabajo Práctico

|  |  |
| --- | --- |
| **Apellido y Nombre** | **Padrón** |
| Kusnesov, Uriel | 93252 |
| Aguila Cudicio, Barbara | 92071 |

CONTENIDO

CONTENIDO 2

OBJETIVO 3

Encadenamiento………………………………………………………………………………………………………………………….3

Encadenamiento hacia adelante 4

Encadenamiento hacia atrás 5

# 

# OBJETIVO

El objetivo del presente trabajo práctico es:

* Continuar con el análisis de los sistemas expertos realizados en la materia correlativa 75.67 Sist. Automáticos de Diagnósticos y detección de Fallas I.
* Profundizar más sobre los sistemas expertos.

# Encadenamiento

Se utilizaron clases auxiliares para representar las reglas y la base de conocimiento.

public class BaseDeConocimiento

{

public IList<Regla> reglas { get; set; }

public IList<string> Hechos { get; set; }

public BaseDeConocimiento(IList<Regla> reglas, IList<string> hechos)

{

this.reglas = reglas;

this.Hechos = hechos;

}

}

public class Regla

{

public IList<string> Condiciones { get; set; }

public string Premisa { get; set; }

public bool Disparada { get; set; }

public Regla(IList<string> condiciones, string premisa)

{

this.Condiciones = condiciones;

this.Premisa = premisa;

this.Disparada = false;

}

}

# 

# Encadenamiento hacia adelante

//Utilizamos la estrategia la primera regla dispara

private static void EncadenamientoHaciaAdelante(BaseDeConocimiento bdc)

{

var reglasParaDisparar = bdc.reglas.Where(x => x.Condiciones.All(y => bdc.Hechos.Contains(y) && !x.Disparada)).ToList();

while(reglasParaDisparar.Count > 0)

{

var regla = reglasParaDisparar.First();

Console.WriteLine("Disparando regla con condiciones" + regla.Condiciones.ToString());

regla.Disparada = true;

bdc.Hechos.Add(regla.Premisa);

Console.WriteLine("Agrego a los hechos la premisa " + regla.Premisa);

Console.WriteLine("Hechos:");

foreach(var hecho in bdc.Hechos)

{

Console.WriteLine(hecho);

}

reglasParaDisparar = bdc.reglas.Where(x => x.Condiciones.All(y => bdc.Hechos.Contains(y) && !x.Disparada)).ToList();

}

Console.WriteLine("No hay mas reglas que disparar");

}

# Encadenamiento hacia atrás

//Utilizamos como estrategia ver si la primera condicion se cumple

private static bool EncadenamientoHaciaAtras(BaseDeConocimiento bdc, string premisaHipotesis)

{

Console.WriteLine("Encadenamiento hacia atras para la hipotesis " + premisaHipotesis);

if (bdc.Hechos.Contains(premisaHipotesis))

{

Console.WriteLine("Hipótesis " + premisaHipotesis + " verificada");

return true;

}

var reglasParaHipotesis = bdc.reglas.Where(x => x.Premisa.Equals(premisaHipotesis)).ToList();

foreach(var regla in reglasParaHipotesis)

{

foreach (var condicion in regla.Condiciones)

{

Console.WriteLine("Verificando Condicion " + condicion);

if (!bdc.Hechos.Contains(condicion))

{

if (EncadenamientoHaciaAtras(bdc, condicion) == false)

{

Console.WriteLine("No se pudo verificar la hipotesis " + premisaHipotesis);

return false;

}

}

Console.WriteLine("Condicion " + condicion + " verificada");

}

}

bdc.Hechos.Add(premisaHipotesis);

Console.WriteLine("Hipótesis " + premisaHipotesis + " verificada");

return true;

}