

# Centro de Enseñanza Técnico Industrial Plantel Colomos



## Reporte 1

Eduardo Camarena Orozco

Matricula: 19310384 Grupo: 7°E1

Ingeniería Mecatrónica

Sistemas Expertos

Mauricio Alejandro Cabrera Arellano

## Teoría

### Sistemas basados en reglas

Basados en reglas previamente establecidas. Los sistemas basados en reglas trabajan mediante la aplicación de reglas, comparación de resultados y aplicación de las nuevas reglas basadas en situación modificada. También pueden trabajar por inferencia lógica dirigida, bien empezando con una evidencia inicial en una determinada situación y dirigiéndose hacia la obtención de una solución, o bien con hipótesis sobre las posibles soluciones y volviendo hacia atrás para encontrar una evidencia existente (o una deducción de una evidencia existente) que apoya una hipótesis en particular.

Representación del conocimiento.

Hay numerosas formas de representar el conocimiento en IA, sin embargo, los Sistemas Expertos suelen ser llamados sistemas basados en reglas.

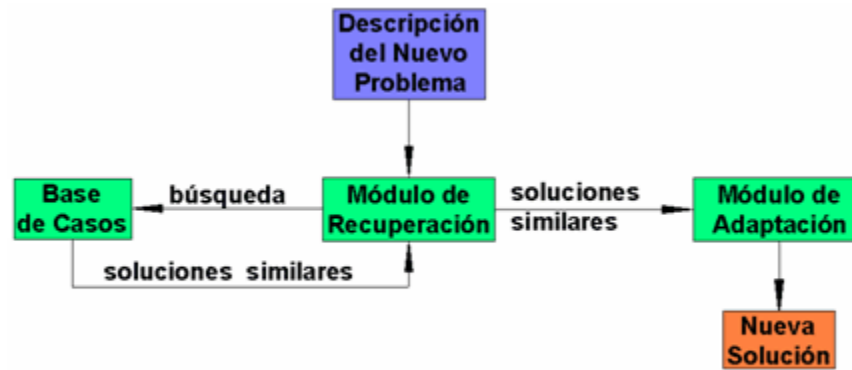
Reglas “Si...entonces...”

Las reglas “si...entonces...” son el principal tipo de conocimiento usado en Sistemas Expertos, donde dichas normas se utilizan para capturar razonamiento de expertos que emplean a menudo. Sin embargo, con el tiempo los investigadores comenzaron a desarrollar e integrar otras formas de representación del conocimiento, tales como el razonamiento basado en casos.

Los sistemas que incluyen múltiples tipos de conocimiento a veces se conocen 356 Ciencia y Tecnología, 13, 2013, pp. 349-364 ISSN 1850-0870S. Badaro, L. J. Ibañez y M. J. Agüero Sistemas Expertos: Fundamentos, Metodologías y Aplicaciones como sistemas híbridos, o etiquetados después de un determinado tipo de representación del conocimiento, por ejemplo, basado en casos (O’Leary, 2008).

### Sistemas basados en casos

El Razonamiento Basado en Casos significa usar viejas experiencias para comprender y resolver nuevos problemas. En él quien razona recuerda una situación previa, similar a la actual y usa esto para resolver el nuevo problema. En general, un caso consiste en la descripción de un problema y la solución dada al mismo.



**Fig. 3.** Estructura de un Sistema Basado en Casos típico

El diseñador a la hora de elaborar un caso primero debe decidir qué lo comprende y conforma, ya que éste no es más que la descripción de un problema y la solución dada al mismo.

Un Sistema Basado en Casos típico consta de 3 partes principales: la Base de Casos, el Módulo de Recuperación y el Módulo de Adaptación. Los casos son problemas resueltos y almacenados en la Base de Casos. Cuando hay un nuevo problema que resolver, éste es descrito para el Módulo de Recuperación, el cual realiza una búsqueda en la Base de Casos y encuentra problemas o casos similares.

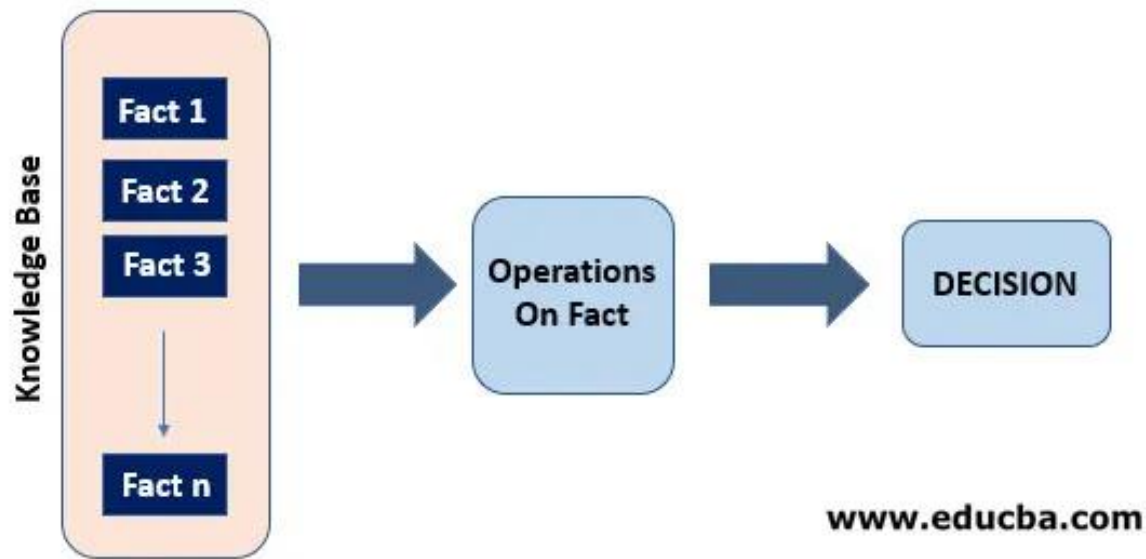
Estos problemas o casos similares resueltos son recuperados (soluciones similares) y enviados al Módulo de Adaptación, donde son analizados para construir una solución para el nuevo problema.

Una vez hallada la solución, se almacena junto con la descripción del problema en la Base de Casos, constituyendo un nuevo caso.

En el establecimiento de los métodos de adaptación se hace imprescindible el conocimiento de los expertos en este dominio. La adaptación de un caso puede ser llevada a cabo por la transformación de un caso solo para ajustarlo a los requerimientos de la nueva situación o mediante la composición apropiada de partes de varios casos. La adaptación a menudo es muy difícil y puede ser eludida del todo, es por esto que ella ha recibido mucha menos atención que la recuperación.

### **Encadenamiento hacia adelante**

El motor de encadenamiento hacia adelante pasa por todos los hechos, condiciones y derivaciones antes de deducir el resultado, es decir, comienza con un conjunto de reglas para realizar una cadena de operaciones para concluir la decisión final. Esta estrategia se utiliza para llegar a la conclusión manipulando el conocimiento de la base de conocimiento.



Esta estrategia se utiliza para responder la pregunta "¿QUÉ PUEDE SUCEDER A CONTINUACIÓN?"

Propiedades:

- Dado que se mueve de arriba a abajo se llama un enfoque de arriba hacia abajo.
- Llega a una conclusión al hacer deducciones de los datos y pasar del estado inicial al estado objetivo.

### Resultados del programa

#### Funcionamiento

Este programa es muy sencillo de usar, aunque no cuenta con manejo de errores, entonces pues a veces se harán cosas que realmente no quisiéramos por errores que se pueden cometer al escribir o algo así, pero pues en general es bastante intuitivo aunque tiene algunas cosas que si no lo haces como deberías no te funcionará de la manera que tú quieres.

Para utilizar el programa primero, si es que no tenemos aún nuestros datos de diferentes personajes, se debe crear manualmente un personaje, esto con el comando `Characters.add('Goku', 'Tu personaje vencio a Freezer en Namek?', 'Sayain Goku')` que lo que hace es agregar un nodo con el nombre de "Goku" y la pregunta que se hará para intentar adivinar quien es, así como el grupo al cual pertenece, ya creado el primer personaje el programa funcionará sin problemas, pero es importante que solo usemos ese comando una vez cuando no tengamos ningún personaje, porque si lo dejamos se seguirá creando ese nodo, el siguiente paso es volver a correr el programa y pensar en un personaje de Dragon Ball, y si no lo tenemos registrado habrá que poner si pertenece a algún grupo de los que nos pregunta o no para saber si se creará un grupo o se agregará un personaje a un grupo ya existente, para ello el programa te pregunta los datos del nuevo personaje y los guarda en el archivo "Personajes", para cada pregunta es importante responder "Si" porque si no respondemos la pregunta de esa forma, si en cambio ponemos "si" no contará como respuesta valida, entonces hay que seguir las indicaciones y si el

programa tiene el personaje que pensamos te dirá el nombre y terminará el programa, funciona de manera muy sencilla pero es eficaz.

Link de GitHub:

<https://github.com/Camarenita/Adivina-Quien.git>

Código:

```
import pickle

class Node:
    def __init__(self,nam,qst, grp):
        self.name = nam
        self.question = qst
        self.group = grp
        self.next = None
        self.previous = None
        self.left = None
        self.rigth = None

class LinkedList:
    def __init__(self):
        self.first = None

        source = open('Personajes', 'ab+')
        source.seek(0)

        try:
            self.first = pickle.load(source)
        except EOFError:
            pass
        finally:
            source.close()
            del source

    def add (self,name,question,group):
        if self.first is None:
            new_node = Node(name,question,group)
            self.first = new_node
            self.saveCharacters()
            return

        current = self.first

        while True:
            if current.group != group:
                if current.rigth == None:
                    new_node = Node(name,question,group)
                    new_node.left = current
                    current.rigth = new_node
                    self.saveCharacters()
                    return
                else:
                    current = current.rigth
            else:
                return
```

```

        while True:
            if current.next == None:
                new_node = Node(name,question,group)
                new_node.previous = current
                current.next = new_node
                self.saveCharacters()
                break
            else:
                current = current.next
        break

def saveCharacters(self):
    files = open('Personajes', 'wb')
    pickle.dump(self.first, files)
    files.close()
    del files

def printCharacters(self):
    current = self.first
    while True:
        print('Nombre: ',current.name, '\nGrupo: ',current.group, '\n')
        if current.next == None:
            while current.previous != None:
                current = current.previous
            if current.rigth != None:
                current = current.rigth
            else:
                break
        current = current.next

def Preguntas(current):
    while True:
        if input('Tu personaje pertenece al grupo de '+current.group+'?\n') == 'Si':
            while True:
                if input(current.question+'\n') == 'Si':
                    print('Tu personaje es ', current.name)
                    return(0)
                else:
                    if current.next == None:
                        return(1)
                    else:
                        current = current.next
            else:
                if current.rigth == None:
                    return(1)
                else:
                    current = current.rigth

def Inicio():
    Characters = LinkedList()

    #Characters.add('Goku','Tu personaje vencio a Freezer en Namek?','Sayain
    Goku')
    #Characters.printCharacters()

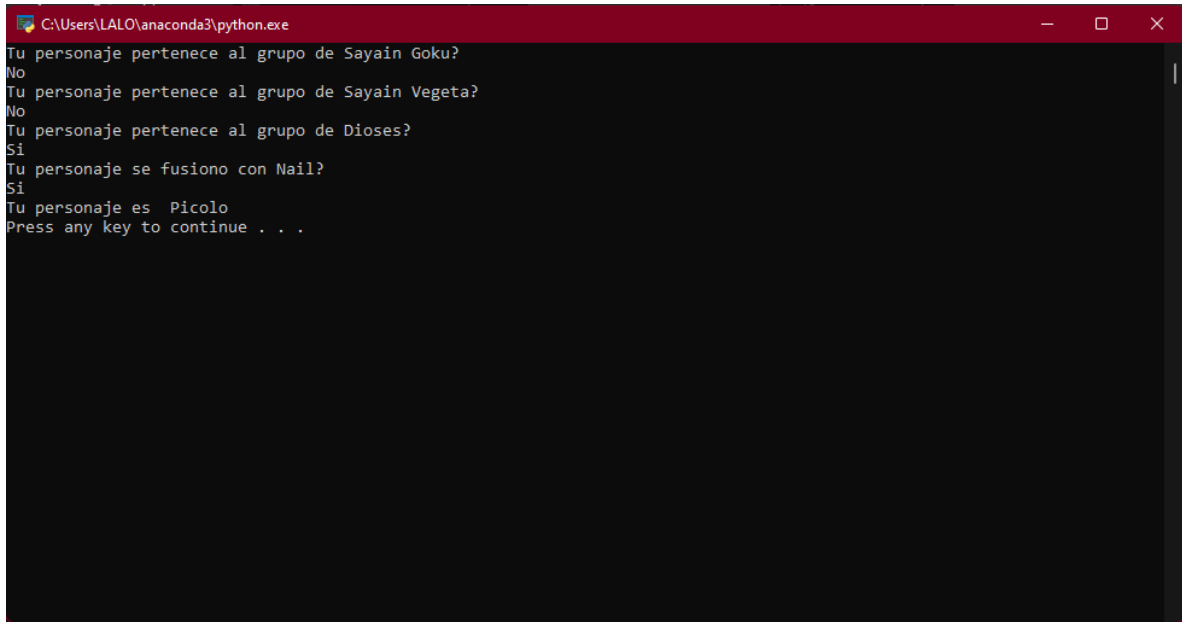
    current = Characters.first

```

```
opc = Preguntas(current)

if opc == 1:
    print('\nAyudame a agregar un nuevo personaje: ')
    name = input('Nombre: ')
    quest = input('Ingrese una pregunta característica para identificar al
personaje: ')
    group = input('Ingrese al grupo al que pertenece: ')
    Characters.add(name, quest, group)
Inicio()
```

Capturas:

A screenshot of a Windows command prompt window titled "C:\Users\LALO\anaconda3\python.exe". The window has a black background with white text. The text shows a series of prompts and user inputs for adding a character. The prompts are: "Tu personaje pertenece al grupo de Sayain Goku?", "Tu personaje pertenece al grupo de Sayain Vegeta?", "Tu personaje pertenece al grupo de Dioses?", "Tu personaje se fusiono con Nail?", and "Tu personaje es Pico". The user inputs are: "No", "No", "Si", "Si", and "Pico". The final prompt is "Press any key to continue . . .".

```
C:\Users\LALO\anaconda3\python.exe
Tu personaje pertenece al grupo de Sayain Goku?
No
Tu personaje pertenece al grupo de Sayain Vegeta?
No
Tu personaje pertenece al grupo de Dioses?
Si
Tu personaje se fusiono con Nail?
Si
Tu personaje es Pico
Press any key to continue . . .
```

```
C:\Users\LALO\anaconda3\python.exe
Tu personaje pertenece al grupo de Sayain Goku?
No
Tu personaje pertenece al grupo de Sayain Vegeta?
No
Tu personaje pertenece al grupo de Dioses?
NO
Tu personaje pertenece al grupo de Tierra?
Si
Tu personaje es un pervertido?
No
Tu personaje es un cerdo?
No

Ayudame a agregar un nuevo personaje:
Nombre: Yamcha
Ingrese una pregunta característica para identificar al personaje: Tu personaje murio contra un sayaman?
Ingrese al grupo al que pertenece: Tierra
Press any key to continue . . . _

C:\Users\LALO\anaconda3\python.exe
Tu personaje pertenece al grupo de Sayain Goku?
No
Tu personaje pertenece al grupo de Sayain Vegeta?
Si
Tu personaje es el principe de los sayains?
No
Tu personaje viajo al pasado para vencer a los androides?
Si
Tu personaje es Trunks
Press any key to continue . . . _
```

## Conclusion

El programa la verdad fue un reto hablando de mis pocos conocimientos sobre el manejo de la memoria estatica, ya que normalmente utilizamos memoria dinamica en la cual podemos jugar con los valores de las variables, eliminarlas y muchas otras cosas más, claro con la desventaja de que al cerrar el programa todo se perdía pero con este programa he aprendido más sobre como manejar variables estáticas en memoria dinamica para poder continuar con la ejecución del programa con las variables creadas anteriormente.

Tiene algunas cosas que mejorarse pero creo que en general el programa funciona bien y cumple bien con su objetivo, tal vez no aprenda automáticamente ya que se le tiene que especificar



diferentes cosas para aprender un nuevo personaje pero en general está muy bien para empezar y ya despues con estos conocimientos apllicarlos y mejorar los futuros programas que haré.

### Referencias

Encadenamiento hacia adelante vs Encadenamiento hacia atrás. (2022). *Encadenamiento hacia adelante vs Encadenamiento hacia atrás / Las 9 principales diferencias para aprender*.

Education-Wiki.com. <https://es.education-wiki.com/4060743-forward-chaining-vs-backward-chaining>

Badaró, S., Javier Ibañez, L., & Agüero, M. (n.d.). *Sistemas Expertos: Fundamentos, Metodologías y Aplicaciones*. [https://www.palermo.edu/ingenieria/pdf2014/13/CyT\\_13\\_24.pdf](https://www.palermo.edu/ingenieria/pdf2014/13/CyT_13_24.pdf)

Moya-Rodríguez, J. L., Becerra-Ferreiro, A. M., & Chagoyén-Méndez, César A. (2012).

Utilización de Sistemas Basados en Reglas y en Casos para diseñar transmisiones por tornillo sinfín. *Ingeniería Mecánica*, 15(1), 01–09.

[http://scielo.sld.cu/scielo.php?script=sci\\_arttext&pid=S1815-59442012000100001](http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1815-59442012000100001)