Programación Lógica

Sergio Camilo Espinosa Botero
Juan Sebastián Reina Zamora
Andrés Felipe Balcero Cerón
Hamed Mohseni

Facultad de INGENIERÍA Sede Bogotá
UNIVERSIDAD NACIONAL DE COLOMBIA

Grandes Paradigmas

Imperativo

- Programador especifica pasos.
- Se define el “cómo”.

Ej. Hacer un sandwich.

1. Tomar pan.
2. Agregar lechuga.
3. ….
4. Agregar jamón.
5. …
6. Terminar con otro pan.

Declarativo

- Programador especifica objetivo.
- Se define el “qué”.

Ej. Hacer un sandwich.

- Hacer un sándwich con jamón, queso, lechuga y tomate.

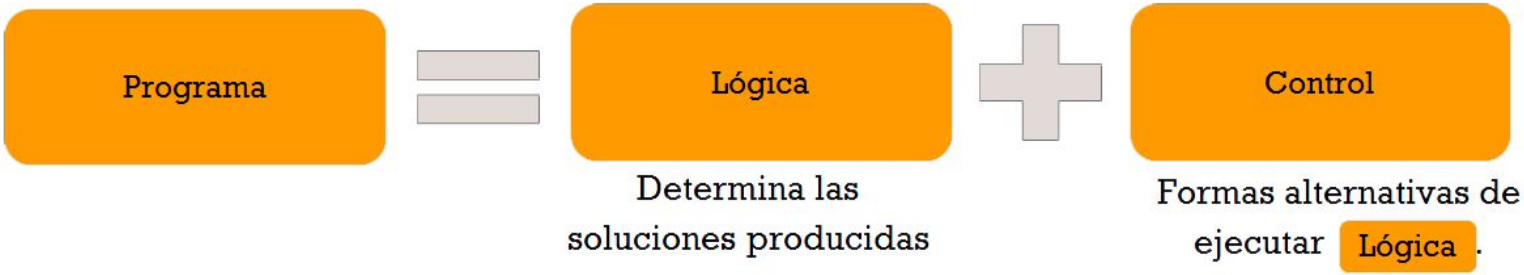
Filosofía de la programación lógica

"Modelar problemas por medio de la abstracción, utilizando un sistema de lógica formal que permite llegar a una conclusión por medio de hechos y reglas"

"Aplicación de reglas de la lógica para inferir conclusiones a partir de datos."

Filosofía de la programación lógica

Se busca representar y ejecutar programas utilizando lógica formal.



Se define un espacio o base de conocimiento utilizando cláusulas, que expresan verdades o hechos del problema a solucionar.

Luego se realizan deducciones o inferencias controladas partiendo de la lógica definida.

Filosofía de la programación lógica

Un problema S a solucionar es un conjunto de reglas y hechos (la lógica), y una solución A es una afirmación que se puede demostrar a partir de la lógica del problema.

Historia de la programación lógica

Conceptos claves

Hecho

Expresión atómica de la forma

P(t1,t2,t3,....,tn)

que verifica la proposición P sobre los objetos t1,t2,t3,…,tn.

“Andrés es estudiante”

Estudiante(Andrés)

Regla

Conjunto de proposiciones lógicas que permiten inferir el valor de verdad de una nueva proposición.

“Todos los estudiantes están matriculados”

Consulta

Proposición construida con el propósito de ser demostrada o de encontrar el conjunto de valores que la convierten en verdadera. Utilizan las reglas definidas anteriormente.

“¿Andrés está matriculado?”

Conceptos claves

Aridad:

Que tantos argumentos u operandos recibe una función, operación o expresión.

- Enunciado - 0. Verdadero
- Propiedad - 1. Andrés es estudiante. Estudiante(Andrés)
- Relación - >1. Andrés es estudiante de la UNAL. Estudiante(Andres, UNAL)

Recursividad:

Especificar una tarea usando la definición de ella misma para un problema más pequeño.

- Caso base.
- Llamado recursivo.

Lógica de Primer Orden

Es un sistema formal diseñado para estudiar la inferencia en los lenguajes de primer orden.

Posee:

- Predicados.
- Cuantificadores.
- Funciones.
- Argumentos.
- Variables y constantes de individuos.

Proposición: Es un enunciado al cual se le puede atribuir un valor de verdad.

Ejemplo:

- Hamed está exponiendo bien.
- Al profesor le está gustando la exposición.

Constantes de individuo: Una constante de individuo es una expresión lingüística que refiere a una entidad.

Ejemplo:

- Hamed
- Carro

Lógica de Primer Orden

Variables de individuo: La lógica de primer orden cuenta con variables, que son expresiones cuya referencia no está determinada.

Ejemplo:

- “Esto es bello” —> Belleza(x)

Cuantificadores: Un cuantificador es una expresión que afirma que una condición se cumple para un cierto número de individuos.

Ejemplo:

- Para todo X, X es ladrón —> $\forall X\ L(X)$
- Existe un X, tal que X no es Ladron —> $\exists X\ L(X)$

Conectivos: La lógica de primer orden incorpora además las conectivas de la lógica proposicional.

Ejemplo: ^,¬,v,=>,<=>

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
falso	falso	verdadero	falso	falso	verdadero	verdadero
falso	verdadero	verdadero	falso	verdadero	verdadero	falso
verdadero	falso	falso	falso	verdadero	falso	falso
verdadero	verdadero	falso	verdadero	verdadero	verdadero	verdadero

Cláusula de Horn

Una fórmula lógica es una cláusula de Horn si es una cláusula (disyunción de literales) con, como máximo, un literal positivo.

$$\neg p \vee \neg q \vee \cdots \vee \neg t \vee u$$

$$(p \wedge q \wedge \cdots \wedge t) \rightarrow u$$

- Cláusula ‘definite’: Cláusula de Horn con exactamente un literal positivo.
- Hecho: Cláusula ‘definite’ sin literales negativos.
- Cláusula objetivo: Sin ningún literal positivo (consulta).

Cláusula de Horn

antecedente → consecuente "Si es verdad el antecedente, entonces es verdad el consecuente"

En Prolog se escribe primero el consecuente y luego el antecedente.

Estructura de cláusulas de Horn:

$$\neg mujer(A) \vee \neg padre(B, A) \vee hija(A, B)$$

$$(mujer(A) \wedge padre(B, A)) \rightarrow hija(A, B)$$

Estructura de cláusulas de Horn en Prolog:

"A es hija de B si A es mujer y B es padre de A"

```
hija(A,B) :- mujer(A), padre(B,A)
```

Resolución SLD

Es la regla de inferencia básica utilizada en programación lógica.

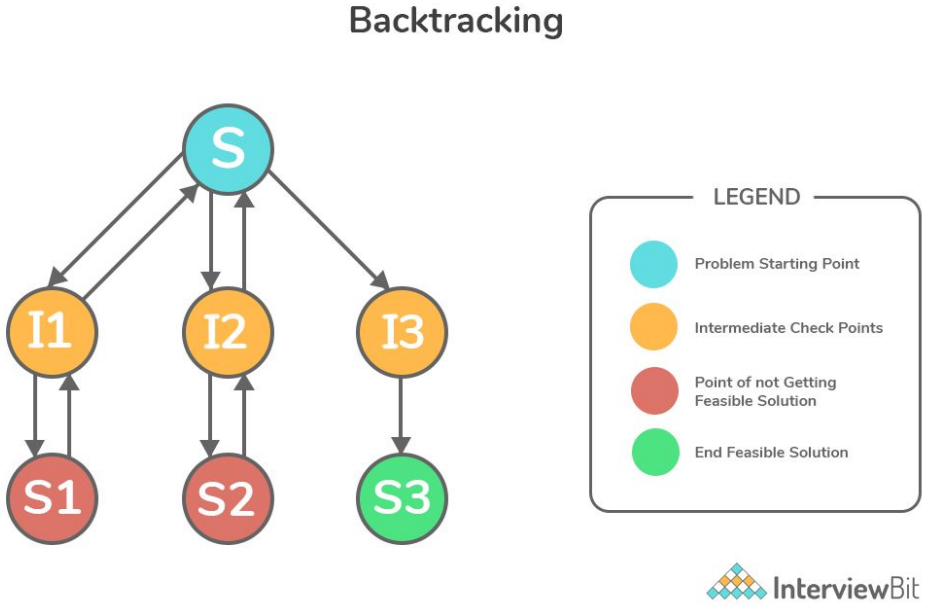
$$C_1, C_2, \cdots, C_l$$

Donde la "cláusula superior" C1, es una cláusula de entrada, y cada otra cláusula Ci+1 es una solución de cuyos padres es la cláusula anterior Ci. La prueba es una refutación si la última cláusula Cl,es la cláusula vacía.

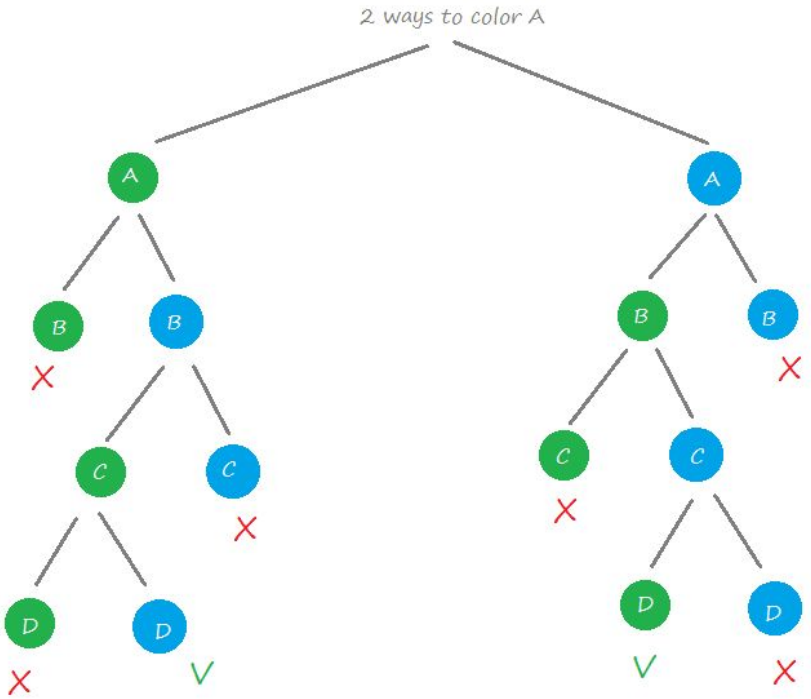
En SLD, todas las cláusulas son una secuencia de cláusulas objetivo y el otro padre es una cláusula de entrada. En la resolución SL, el otro padre es una cláusula de entrada o una cláusula ancestral anterior en la secuencia.

Backtracking

Es una técnica algorítmica para solución de problemas de manera recursiva. Se basa en buscar una solución analizando todas las opciones disponibles.



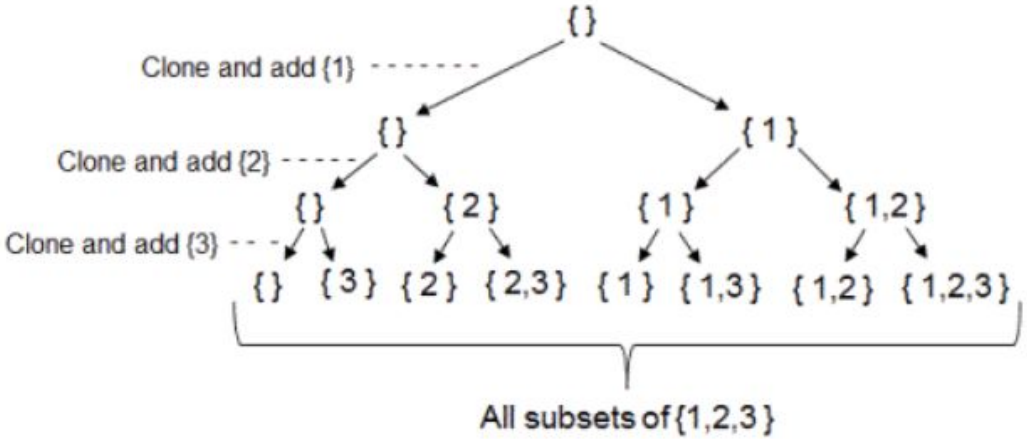
Backtracking



Si una opción no es válida el algoritmo de backtracking se devuelve hasta el anterior punto de elección y toma el siguiente camino.

Sigue así hasta encontrar la solución.

Backtracking: Subconjuntos.



Un ejemplo que usa el backtracking es la búsqueda de subconjuntos para un conjunto determinado.

En este algoritmo, en cada punto de elección se da la opción de sí agregar el ‘i-ésimo” elemento o no.

Así se pueden armar todos los 2^n subconjuntos para un conjunto de n elementos

Backtracking: Subconjuntos

Pseudocódigo

```
1   int arr[N]
2  void allSubsets(int pos, int len, int[] subset) {
3    if (pos == N) //position
4    {
5      print(subset)
6      return
7    }
8    subset[len] = arr[pos]
9    //take element
10   allSubsets(pos + 1, len + 1, subset)
11   // Skip the current element.
12   allSubsets(pos + 1, len, subset)
13 }
```

Backtracking: Sudoku

Otro ejemplo de algoritmo de backtracking es la solución de un sudoku. Esta animación muestra cómo se va llenando el sudoku, colocando números para cumplir las condiciones del problema, y devolviéndose a un punto anterior si llega a un momento en el que las condiciones no se cumplen.

5	3	1	2	7	6	8	9	4
6	2	4	1	9	5	2		
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Ventajas

- La programación lógica se puede utilizar para expresar conocimiento de una manera que no depende de la implementación, haciendo que los programas sean más flexibles, comprimidos y comprensibles.
- Permite separar el conocimiento del uso, por ejemplo, se puede cambiar la arquitectura de la máquina sin cambiar los programas o su código subyacente.
- Puede modificarse y extenderse de manera natural para apoyar formas especiales de conocimiento, tales como meta-nivel o conocimiento de orden superior.
- Se puede utilizar en disciplinas no computacionales basadas en el razonamiento y en medios precisos de expresión.
- Expresión simple y precisa de los problemas.
- Generación rápida de prototipos e ideas complejas.
- Sencillez en la implementación de estructuras complejas.
- Potencia.

Desventajas

- No hay una forma adecuada de representar conceptos computacionales encontrados en los mecanismos incorporados de las variables de estado (como se suele encontrar en los lenguajes convencionales).
- Inicialmente, debido a una inversión insuficiente en tecnologías complementarias, los usuarios fueron mal atendidos.
- Al principio, instalaciones deficientes para soportar aritmética, tipos, etc. tenían un efecto desalentador en la comunidad de programación.
- Algunos programadores siempre tienen, y siempre preferirán la naturaleza abiertamente operativa de los programas operados por máquinas, ya que prefieren el control activo sobre las "partes móviles".
- Dependiendo del problema a solucionar, la implementación y el motor de inferencia, puede llegar a ser extremadamente ineficiente.
- Pocas y muy específicas áreas de aplicación.
- Existen muy pocas herramientas de depuración, en su mayoría poco efectivas.
- Es poco utilizado.
- Si el programa no contiene suficiente información para responder una consulta la respuesta puede ser una que se preste para malentendidos.
- Inferencia limitada por su base de conocimiento.

Datalog

Basado en Prolog, es un lenguaje de programación lógica declarativo en el qué:

- Cada fórmula es una cláusula de Horn libre de funciones
- Cada variable en la cabeza de la cláusula debe aparecer en el cuerpo de la misma. Una variable debe comenzar con una letra mayúscula

Es un sistema para bases de datos deductivo en el que las actualizaciones a una BD son escritas en lenguaje lógico.

El simbolo `:-` separa la cabeza de la clausula de las reglas.

```
parent(john, douglas)
    ancestor(A, B) :-
        parent(A, B)
    ancestor(A, B) :-
        parent(A, C),
        ancestor(C, B)
```

Datalog

Los hechos se guardan en tablas. Si el nombre de una tabla es “padre” y Anakin es el padre de Luke, se almacena este hecho en la base de datos así:

```
> padre(Anakin, Luke).
```

Se pueden hacer búsquedas para ver si una línea está en la tabla. Para eso se pone un ? al frente de la frase:

```
> padre(Anakin, Luke)?
```

```
padre(Anakin, Luke).
```

Si realizo una búsqueda que no se haya puesto:

```
> padre(Anakin, Han)?
```

No va a arrojar resultados

Si añadimos más líneas:

```
> padre(Anakin, Leia).
```

```
> padre(Tywin, Tyrion).
```

```
> padre(James, Harry).
```

Podemos buscar todos los hijos de Anakin:

```
> padre(Anakin, B)?
```

```
padre(Anakin, Luke).
```

```
padre(Anakin, Leia).
```

Datalog

Se pueden también crear nuevas reglas para las búsquedas:

```
> ancestor(A, B) :- padre(A, B).

> ancestro(A, B) :- padre(A, C), ancestro(C, B).
```

Luego agrego más datos:

```
>padre(Schmi,Anakin).

>padre(Leia,Ben).
```

Después puedo consultar por los ancestros de alguien:

```
>ancestro(A,Ben)?

  ancestro(Schmi, Ben).

  ancestro(Anakin, Ben).

  ancestro(Leia, Ben).
```

Ejemplos Prolog

A continuación un pequeño programa ejemplo en prolog en el que se definen hechos, reglas y consultas.

```
1 dog(fido).
2 dog(rover).
3 dog(henry).
4 cat(felix).
5 cat(michael).
6 cat(jane).
7 animal(X) :- dog(X).
```

HECHOS

REGLAS

```
?- animal(fido).
true
```

```
?- animal(felix).
false
```

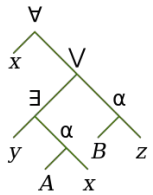
CONSULTAS

Ejemplo Unificación

En prolog y otros lenguajes de programación lógicos existe el concepto de unificación, que consiste en LIGAR las variables a valores, también es llamado matching en otras fuentes.

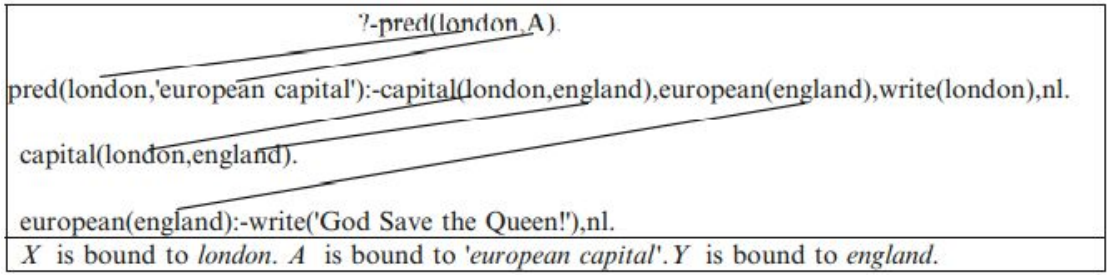
Mediante la unificación y el backtracking es posible entonces la realización de varias consultas y la programación lógica subyacente.

$$\forall x \left((\exists y \, A(x)) \vee B(z) \right)$$




```
1 capital(london,england).
2 european(england):-write('God Save the Queen!'),nl.
3 pred(X,'european capital') :- capital(X,Y),european(Y),write(X),nl.
```

```
?- pred(london,A).
God Save the Queen!
london
A = 'european capital'
```



Casos de uso

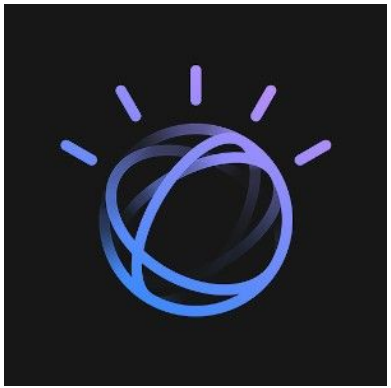
Dentro de los casos de uso, podemos observar su uso en los siguientes proyectos:

-TerminusDB: Esta es una base de datos controlada por modelos, capaz de proporcionar muchas de las características y comportamientos que Git hace para la gestión de versiones, pero para grandes conjuntos de datos.

-Watson de IBM: Asistente de humanos expertos

-GeneXus: plataforma de desarrollo que usa Prolog para transformar y traducir sus funcionalidades deseadas en código.

TerminusDB



GeneXus™

Aplicaciones.

El uso del paradigma de programación lógica tiene como algunas aplicaciones las siguientes:

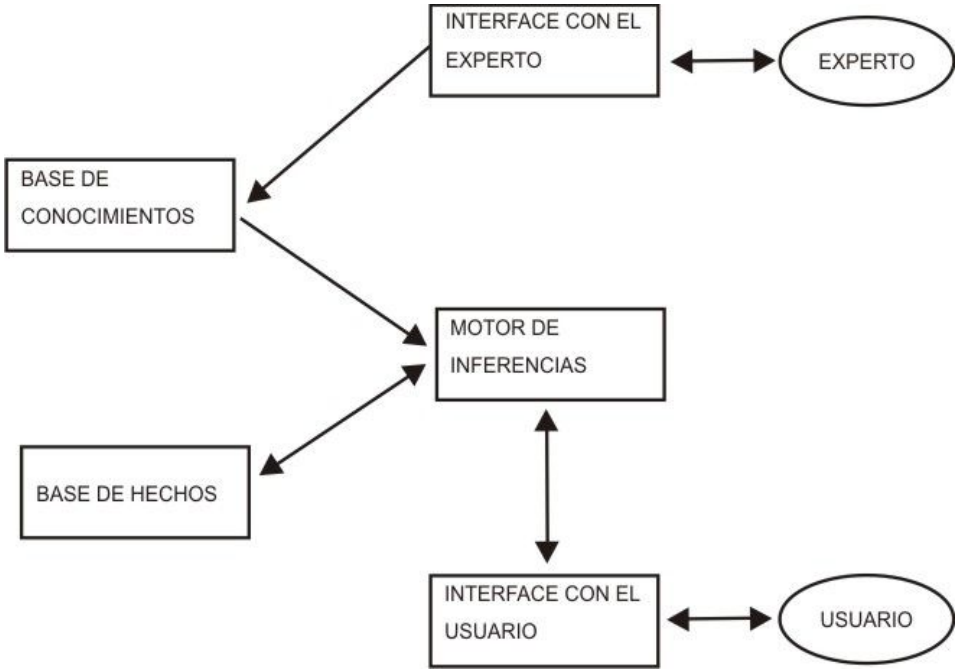
- Análisis de oraciones en un lenguaje natural dada su gramática (poder realizar acciones como preguntar el significado de una palabra, traducirlo a otro idioma).
- Análisis y medición de redes sociales
- Construcción de Sistemas Expertos
- Sistemas de asesoramiento jurídico
- Sistemas de apoyo a los médicos
- Estandarizar la representación del conocimiento.
- Sistemas basados en agentes y búsquedas o IA

Sistemas Expertos

Son sistemas basados en computadoras, interactivos y confiables, que pueden tomar decisiones y resolver problemas complejos.

El software de IA tiene el suficiente conocimiento almacenado como para resolver problemas complejos que solo un experto humano podría resolver.

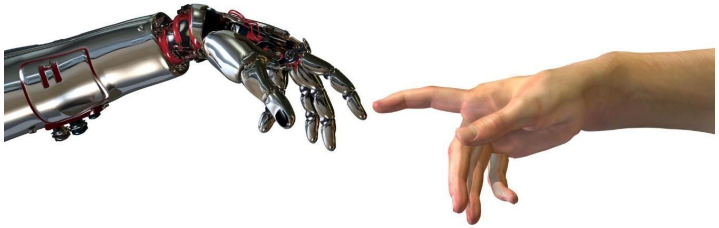
Debido a que los sistemas expertos usan hechos y reglas como base de conocimiento, la programación lógica es muy usada para su desarrollo.



Sistemas expertos.

Algunos ejemplos de sistemas expertos son:

- PXDES: Determina el grado de cáncer de pulmón.
- CADET: Identifica cáncer en primeras etapas.
- DENDRAL: Puede predecir la estructura molecular, basado en los datos espectrográficos de una sustancia.
- DXPlain: Sugiere varias enfermedades a partir de un cuadro clínico.



Otras aplicaciones:

- Útil para proyectos de reparación y mantenimiento.
- Optimización de almacenes.
- Planificación y programación.
- La configuración de objetos fabricados.
- Toma de decisiones financieras Publicación de conocimiento.
- Monitorización y control de procesos.
- Bolsa de comercio.
- Horarios de aerolínea y horarios de carga.

Procesamiento del lenguaje natural

Es una disciplina enfocada en la comprensión, manejo y generación del lenguaje natural pero gestionado por las máquinas o computadoras. Por lo tanto se basa en la capacidad de que una computadora se comunique con humanos.

La programación lógica fue aplicada aquí en los métodos basados en reglas, definimos las reglas que permiten procesar el lenguaje natural deseado mediante una gramática, posteriormente fueron superados con técnicas de aprendizaje de máquina debido a la complejidad y tamaño de las reglas.

Demostración Automática de Teoremas

Es un subcampo desarrollado del razonamiento automático y consiste en demostrar teoremas matemáticos mediante programas de computador.

Se considera que un proceso demuestra un teorema si este comienza con axiomas y se van produciendo nuevos pasos aplicando reglas de inferencia, sin embargo es uno de varios métodos que permiten resolver este problema.

Se utiliza en diseño y verificación de circuitos integrados.

$$\begin{array}{l} p \rightarrow q \\ q \rightarrow r \\ \hline \therefore\ p \rightarrow r \end{array}$$

Referencias

- Use Backtracking Algorithm to Solve Sudoku - DEV Community
- Backtracking | Introduction - GeeksforGeeks
- Print all Subset for set (Backtracking and Bitmasking approach) (topcoder.com)
- http://inteligenciaartificialgrupo33.blogspot.com/p/sistemas-expertos-los-sistemas-expertos.html
- http://ferestrepoca.github.io/paradigmas-de-programacion/proglogica/logica_teoria/index.html
- Sistemas Expertos: Definición, Aplicaciones y Ejemplos (tecnologias-informacion.com)
- ¿Qué son los Sistemas Expertos (SE)? - Solvenup
- https://datascientest.com/es/nlp-natural-language-processing-introduccion
- 2 Tutorial (racket-lang.org)
- https://cibernetico.org/2015/07/21/cuales-son-los-componentes-de-un-sistema-experto/
- https://www.wiki.es-es.nina.az/Demostraci%C3%B3n_autom%C3%A1tica.html
- https://rpubs.com/Carlos_Angulo/Conteo
- https://es.wikipedia.org/wiki/Anexo:Reglas_de_inferencia#Reglas_de_la_l%C3%B3gica_de_predicados_cl%C3%A1sica
- https://betterprogramming.pub/have-you-heard-of-prolog-ce34fdb8660a
- https://es.wikipedia.org/wiki/Variable_libre_y_variable_ligada
- Brammer Max,Logic Programming with Prolog.

MUCHAS GRACIAS POR SU ATENCIÓN!!!