UAEM

UAP NEZAHUALCÓYOTL

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

UNIDAD ACADÉMICA PROFESIONAL NEZAHUALCÓYOTL

LICENCIATURA EN INGENIERÍA EN SISTEMAS INTELIGENTES

MANUAL PARA PRÁCTICAS DEL LABORATORIO DE CÓMPUTO PARA LA ASIGNATURA DE INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL

ELABORARÓN:

DRA. DORICELA GUTIERREZ CRUZ
M. en C. YAROSLAF AARÓN ALBARRÁN FERNÁNDEZ
DR. RICARDO RICO MOLINA

UAEM
UNIDAD ACADÉMICA PROFESIONAL
UAEM NEZAHUALCÓYOTL

MANUAL PARA PRÁCTICAS DEL LABORATORIO DE CÓMPUTO PARA LA ASIGNATURA DE INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL

IDENTIFICACIÓN DE LA UNIDAD DE APRENDIZAJE

ESPACIO ACADÉMICO : Unidad Académica Profesional - Nezahualcóyotl							
PROGRAMA EDUCATIVO: **LICENCIATURA EN INGENIERÍA EN SISTEMAS INTELIGENTES**					**Área de docencia: : INGENIERÍA Y TECNOLOGÍA**		
Aprobación por los H.H. Consejos Académico y de Gobierno		**Fecha:** AGOSTO 2016			Programa elaborado por: **Dra. Doricela Gutiérrez Cruz** **Ing. Yaroslaf Aarón Albarrán Fernández**		
Nombre de la Unidad de Aprendizaje: INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL					**Fecha de elaboración:** 11-08-2016		
Clave L40635	**Horas de teoría** 2	**Horas de práctica** 1	**Total de horas** 3	**Créditos** 4	**Tipo de Unidad de Aprendizaje** Curso	**Carácter de la Unidad de Aprendizaje** Obligatoria	**Núcleo de formación** Integral
Prerrequisitos Contar con conocimientos básicos de lógica matemática.		**Unidad de Aprendizaje Antecedente** Lógica matemática			**Unidad de Aprendizaje Consecuente** Introducción al tratamiento de imágenes		

UAEM
UNIDAD ACADÉMICA PROFESIONAL
UAEM NEZAHUALCÓYOTL

RECIBIDO
10 OCT 2016
SUB. ACADÉMICA
UAP Nezahualcóyotl

ÍNDICE

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

DIRECTORIO

Dr. en D. Jorge Olvera García
Rector

Dr. en Ed. Alfredo Barrera Baca
Secretario de Docencia

Dra. en Est. Lat. Ángeles Ma. del Rosario Pérez Bernal
Secretaria de Investigación y Estudios Avanzados

Dr. en D. José Benjamín Bernal Suárez
Secretario de Rectoría

Mtra. en E. P. D. Ivett Tinoco García
Secretaria de Difusión Cultural

Mtro. en C. I. Ricardo Joya Cepeda
Secretario de Extensión Vinculación

Mtro. en E. Javier González Martínez
Secretario de Administración

Dr. en C. Pol. Manuel Hernández Luna
Secretario de Planeación y Desarrollo Institucional

Mtra. en A. Ed. Yolanda E. Ballesteros Sentíes
Secretaria de Cooperación Internacional

Dr. en. D Hiram Raúl Piña Libien
Abogado General

Lic. en Com. Juan Portilla Estrada
Director General de Comunicación Universitaria

Lic. Jorge Bernáldez García
Secretario Técnico de la Rectoría

Mtro. en A. Emilio Tovar Pérez
Director General de Centros Universitarios y Unidades Académicas Profesionales

Mtro. en A. Ignacio Gutiérrez Padilla
Contralor

UNIDAD ACADEMICA PROFESIONAL NEZAHUALCOYOTL

DIRECTORIO

M. En E.U.R. Héctor Alanís
Coordinador

Dr. Darío Ibarra Zavala
Dr. en Subdirector Académico

Lic. en E. Alfredo Ríos Flores
Subdirector Administrativo

Dra. en C. S. María Luisa Quintero Soto
Coordinadora de Investigación y Estudios Avanzados

Lic. en A. E. Víctor Manuel Durán López
Coordinador de Planeación y Desarrollo Institucional

Dr. en R.I. Rafael Alberto Duran Gómez
Coordinador de la Licenciatura en Comercio Internacional

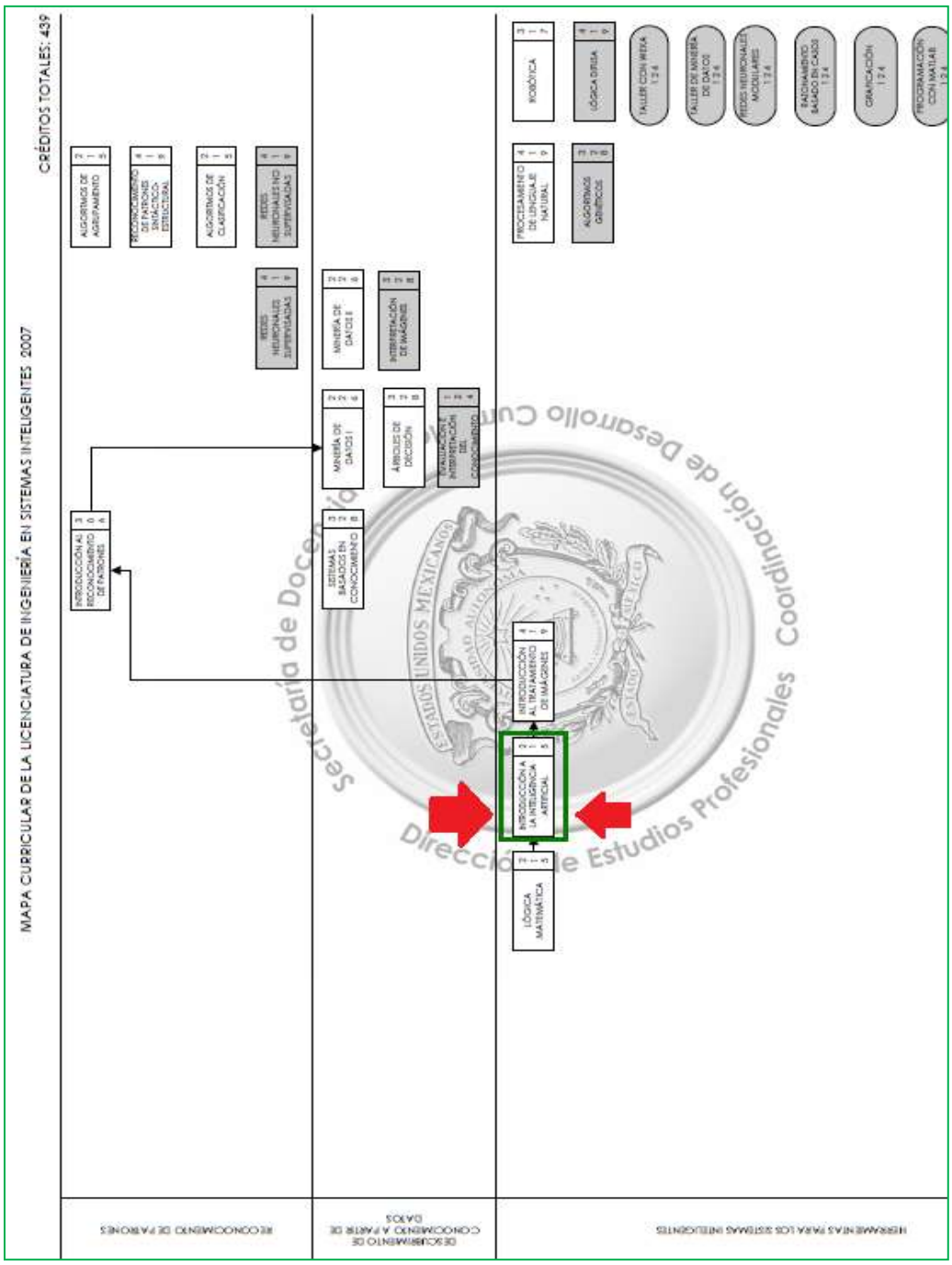
Dra. en C. Georgina Contreras Landgrave
Coordinadora de la Licenciatura en Educación para la Salud

Dra. en S. Ricardo Rico Molina
Coordinador de la licenciatura en Ingeniería en Sistemas Inteligentes

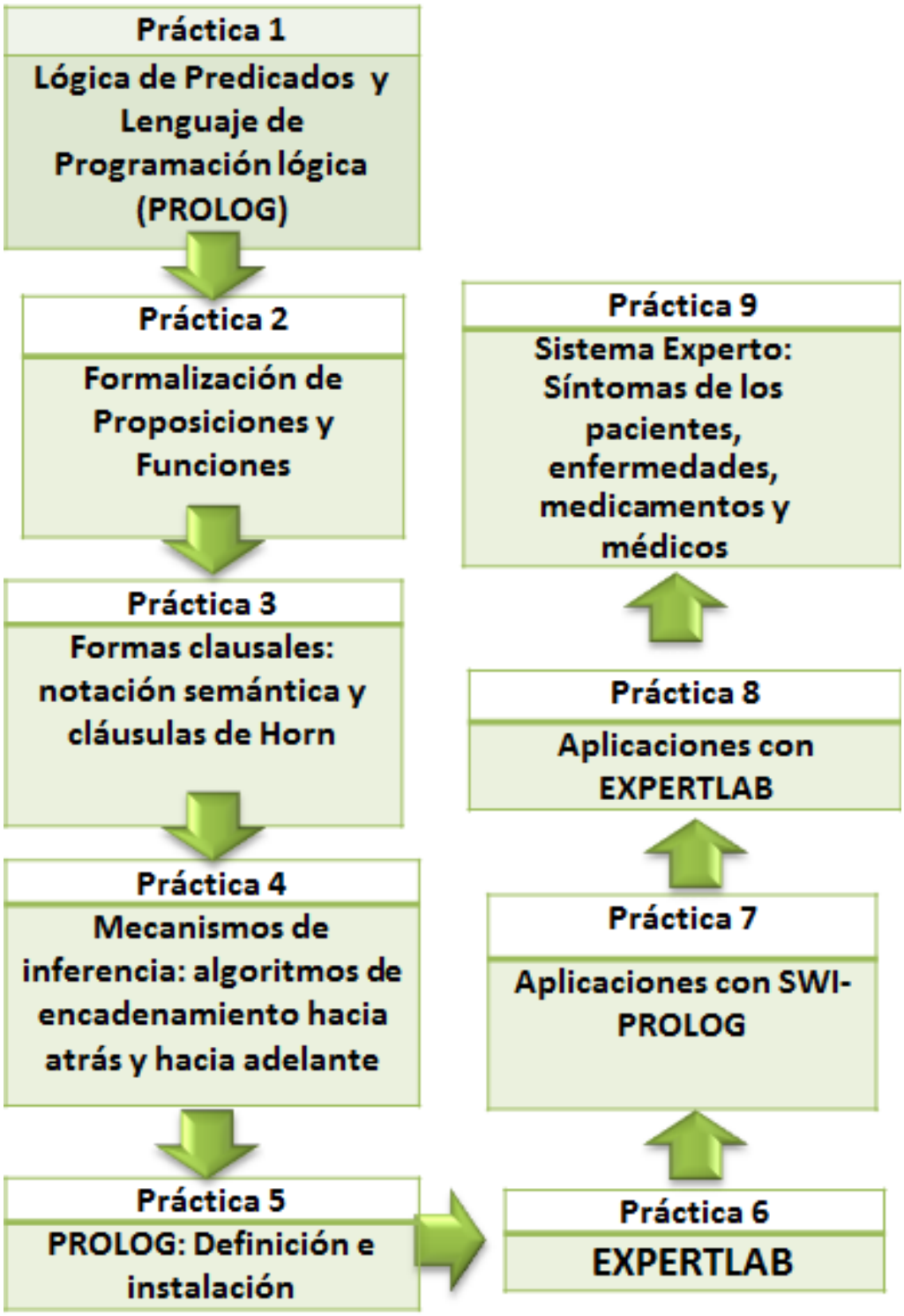
D. En U. Noé Gaspar Sánchez
Coordinador de Ingeniería en Transporte

M. En CC Erick Nicolás Cabrera Álvarez
Coordinador de Licenciatura en Seguridad Ciudadana

Ubicación de la asignatura de Introducción a la Inteligencia Artificial, dentro del programa de la Lic. en Ing. en Sistemas Inteligentes.



SECUENCIA DIDÁCTICA



PRÁCTICA 1

Lógica de Predicados: un acercamiento al Lenguaje de Programación lógica (PROLOG)

OBJETIVO

- Analizar de forma interna la estructura de las oraciones lógicas que permitan en lo consecuente introducirlas en el lenguaje de programación lógica.

INTRODUCCIÓN

La lógica es un lenguaje que permite expresar conocimiento y razonar a partir de ciertas expresiones deducir otras (deducción), sus principales características son conformadas por: sintaxis y semántica bien definidas, así como también reglas de inferencia.

La lógica de predicados constituye una extensión de la lógica. Esta extensión es explicita y sistematiza el proceso inferencial que se efectúa cuando se trabaja con funciones proposicionales y cuantificadores, o de manera más precisa, con un *lenguaje de primer orden*. La lógica de predicados se emplea como una técnica de especificación formal dentro de la metodología de la programación; esto es: se traducen las especificaciones de los programas a un lenguaje de primer orden con objeto de que las reglas formales del mismo pongan de manifiesto posibles fallos del soporte lógico, intentando así aumentar la fiabilidad del diseño del programa. Estos *métodos formales* son una técnica más de la programación lógica.

La lógica de predicados refleja aspectos de estructura gramatical que representan la estructura de cualquier oración del lenguaje natural, por lo que resulta de suma importancia conocer los límites de expresividad y el estudio de alfabeto y de silogismos que la componen.

El alfabeto de la lógica de predicados contiene principalmente cinco clases de símbolos, como son:

(1) **conectivos lógicos:** $\rightarrow, \wedge, \vee, \neg, \forall, \exists, =$

(2) **Símbolos *de* predicado:** $P_1^2, P_2^4, ..., Q^n, R^m, S^k$ (se utilizan usualmente las letras del medio del alfabeto en mayúscula)

(3) **Constantes:** $a_1, a_2, ..., b, c, d$ (se utilizan usualmente las primeras letras del alfabeto en minúscula)

(4) **Variables:** $x_1, x_2, ..., y, z$ (se utilizan usualmente las últimas letras del alfabeto en minúscula, las variables usuales)

(5) **Símbolos auxiliares:** (,)

Entre los conectivos lógicos hay tres símbolos nuevos que de alguna forma se adicionan a la simbología utilizada en la lógica proposicional, estos son:

Cuantificador universal $\forall$

Permite reflejar la estructura de oraciones, ejemplo: "todos los hombres son fieles"; en general expresa algo sobre el conjunto de cosas: e.g. "los hombres"

Cuantificador existencial $\exists$

Permite reflejar la estructura de oraciones como: "algún hombre es esclavo"; representan la existencia de un determinado objeto.

Cuantificador de igualdad =

Es una relación que se da entre objetos y permite representar oraciones como: "Peter Parker y Spiderman son la misma persona".

P_1^2 Dentro de la simbología anteriormente expuesta; también hay relaciones como "ser mayor que", "estar a mitad de"; reflejados por los subíndices; los cuales establecen la **aridad** de un predicado. Esta aridad de un predicado como "ser fieles" es 1 (predicado unario), pues ser fiel, se predica de un solo objeto, mientras que la aridad de un predicado como "ser mayor que" es 2 (predicado binario) pues se necesitan dos objetos para ser comparados bajo esta relación.

Las **constantes** se utilizan exclusivamente para representar objetos individuales, evidentemente no representan conjuntos o grupos de conjuntos.

Las **variables** de la lógica de predicados son variables de objetos. Una variable como x varía entre objetos particulares pero no entre conjuntos o grupos de cosas.

Reglas de Formación de Fórmulas bien Formadas (fbf)

Además de los símbolos anteriormente mencionados se utilizan **Variables predicativas**, que representan predicados indeterminados. Se usan estas letras mayúsculas: F, G, H.

Las formulas bien formadas en la lógica de predicados de expresarán como **(fbf)** las cuales se dividen en atómicas y complejas.

Formulas atómicas:

Sean $t_1, \ldots . t_n$ términos y ***P*** un símbolo de predicado de *aridad* n. Así, las siguientes son formulas bien formadas:

$$t_1 = t_2$$
$$P(t_1, \ldots, t_n)$$

Formulas complejas:

Sea x una variable. Suponga que φ y ψ son *fbf*. Así, las siguientes son formulas bien formadas:

$$(\phi \rightarrow \psi)$$
$$(\phi \wedge \psi)$$
$$(\phi \vee \psi)$$
$$\neg\phi$$
$$\forall x\phi$$
$$\exists x\phi$$

REGLAS

Regla 1. Cada variable predicativa seguida de una o más constantes individuales es una proposición atómica. *Ejemplos: a) Fa; b) Gab; c) Habc*

Regla 2. Cada proposición atómica afectada al menos por un operador es una proposición molecular. *Ejemplos: a) Fa ∧Gb; b) Fa →(Gb∨Hc); c) Fa ∧Gb∧Hc*

Regla 3. Cada variable predicativa seguida de una o más variables individuales es una función proposicional atómica. *Ejemplos: a) Fx; b) Gxy; c) Hxyz*

Regla 4. Cada función proposicional atómica afectada al menos por un operador es una función proposicional molecular. *Ejemplos: a) Fx∧Gy; b) Fx→(Gy∨Hz); c) Fx∧Gy∧Hz*

Regla 5. Son variables libres las variables que no son afectadas por algún cuantificador.

Ejemplos: a) Fx; b) (Fx→Gy)∨Hz; c) Fx ∧(Gy∧Hz)

Regla 6. Son variables ligadas las variables afectadas por algún cuantificador. Ejemplos:

$$a)\,(\exists x)Fx\ ,\ b)\,(\exists x)(\exists y)(Fx \wedge Gy)\ ,\ c)\,(\forall x)(\forall y)(\forall z)[(Fx \rightarrow Gy) \vee Hz]$$

Regla 7**.** Son fórmulas cerradas las fórmulas que no contienen variables libres.

Ejemplos:

$$a)\,(\exists x)Fx\ ,\ b)\,(\exists x)(\exists y)(Fx \wedge Gy)\ ,\ c)\,(\forall x)(\forall y)(\forall z)[(Fx \rightarrow Gy) \vee Hz]$$

Regla 8. Son fórmulas abiertas las fórmulas que contienen al menos una variable libre.

Ejemplos:

$$a)\ Fx\ ,\ b)\,(\forall x)(\forall y)(\forall z)(Fx \rightarrow Gy) \vee Hz\ ,\ c)\,Fx \wedge (\exists y)(\exists z)(Gy \wedge Hz)$$

Regla 9. Si cuantificamos las variables libres de una función proposicional obtenemos una proposición.

Ejemplos:

$$a)\,Fx:\quad (\forall x)Fx\ ,\ b)\,(Fx \rightarrow Gy) \vee Hz:\quad (\forall x)(\exists y)(\forall z)[(Fx \rightarrow Gy) \vee Hz]$$

Regla 10. Si sustituimos las variables libres de una función proposicional por constantes individuales obtenemos una proposición.

Ejemplos: $a)\,Fx:\ Fa\ ,\ b)\,Gxy:\ Gab\ ,\ c)\,Hxyz:\ Habc$

Regla 11. En la lógica de predicados de primer orden se cuantifican sólo las variables individuales.

Ejemplos: $a)\,(\exists x)Fx\ ,\ b)\,(\exists x)(\exists y)(Fx \wedge Gy)$

DESARROLLO

Realizar formalizaciones bien formadas basándose en las reglas anteriormente expuestas, el alumno deberá desarrollar cinco ejemplos de cada una de las once reglas, indicando el contenido de las variables o constantes que use.

Regla	Ejemplos
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Referencias

[1] S. Russell y P. Norvig, "Inteligencia Artificial. Un enfoque moderno". Prentice Hall, 1996

[2] E. Rich y K. Knight. "Inteligencia Artificial". McGraw-Hill, 1994 (lógica)

[3] N. Nilsson, "Inteligencia Artificial". Una nueva síntesis, McGraw-Hill, Madrid, 2000

[4] Mc Allister, j. "Inteligencia Artificial y Prolog". Editorial Alfa Omega/ Marcombo. México. 1999.

PRÁCTICA 2
Formalización de Proposiciones y Funciones

OBJETIVO

- El alumno conocerá la formalización de funciones y proposiciones mediante transformaciones y cuantificaciones.
- Desarrollará proposiciones singulares, proposicionales, cuantificadas y complejas aplicando las reglas dispuestas para ello, mismas que le servirán en el empleo del lenguaje lógico programable.

INTRODUCCIÓN

La Lógica de Primer Orden analiza las frases sencillas del lenguaje (fórmulas atómicas o elementales) separándolas en Términos y Predicados. Los términos hacen referencia a los objetos que intervienen y los predicados a las propiedades o relaciones entre estos objetos. Además, dichas fórmulas atómicas se pueden combinar mediante Conectivas permitiendo construir fórmulas más complejas, llamadas fórmulas moleculares.

Formalización de proposiciones singulares

Una proposición predicativa se simboliza funcionalmente invirtiendo el orden de sus elementos y, por razones operativas, se usa cualquier letra mayúscula para los predicados y cualquier letra minúscula para las constantes individuales.

Ejemplos:

a) David es abogado: Ad
b) David y Goliat no son médicos: ~ Md ∧ ~ Mg
c) Es falso que David y Goliat sean filósofos: ~ (Fd ∧ Fg)
d) David y Goliat son hermanos: Hdg

Formalización de funciones proposicionales

Las siguientes expresiones '*x es inteligente*' e '*y es sabio*', que se simbolizan respectivamente 'Ix' y 'Sy', son funciones proposicionales, es decir, casi proposiciones ya que sus argumentos están representados por variables que significan individuos indeterminados, de manera que no pueden ser calificadas de verdaderas ni de falsas.

Cuantificación

Una función proposicional expresa simbólicamente la forma de una proposición individual. Para ampliar su significación a más individuos se les anteponen los cuantificadores. Así se tiene:

Px : x es periodista
$(\exists x)$ Px : algunos x son periodistas
$(\forall x)$ Px : todos los x son periodistas

Transformación de Funciones en Proposiciones

Sustituyendo la variable individual por una constante individual.
Ejemplo:
Fx (función proposicional)
Fa (proposición)

Anteponiendo un cuantificador a la Función.
Ejemplo:
Fx (función proposicional)
$(\exists x)$ Fx (proposición)
$(\forall x)$ Fx (proposición)

Formalización de Proposiciones cuantificadas

Las proposiciones cuantificadas, como existenciales y universales, pueden ser afirmativas o negativas, su expresión simbólica es la siguiente:

Proposiciones	Fórmulas
Todo x enseña :	$(\forall x)$ Ex
Ningún x enseña :	$(\forall x) \sim$ Ex
Algún x enseña :	$(\exists x)$ Ex
Algún x no enseña :	$(\exists x) \sim$Ex

Formalización de Proposiciones Complejas

Habiendo establecido la representación funcional de las proposiciones atómicas es posible ahora conectarlas con los operadores de la lógica proposicional para formar con ellas proposiciones moleculares. Por ejemplo:

La noticia es sensacional y **el público aplaude** → $Sn \wedge Ap$
p *q*

Si la función tiene éxito, el promotor se alegra → $Ef \rightarrow Ap$
p *q*

Formalización de Proposiciones Categóricas

Proposición Universal Afirmativa (A)

Todos los limeños son peruanos → Para todo x, si x es limeño, entonces x es peruano
$(\forall x)$ Lx → Px

Proposición Universal Negativa (E)

Ningún Congresista es adolescente → Para todo x, si x es congresista, entonces x no es adolescente
$(\forall x)$ Cx → $\sim$Ax

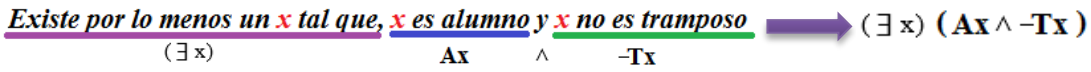
es decir, la fórmula resultante es: $(\forall x)(Cx \rightarrow \sim Ax)$

Proposición Particular Afirmativa (I)

Algunos alumnos de la UAPN provienen del DF

Proposición Particular Negativa (0)

Algunos alumnos no son tramposos



DESARROLLO

Realizar la formalización de proposiciones basándose en las reglas anteriormente expuestas, el alumno deberá desarrollar cinco ejemplos de cada una de ellas, indicando el contenido de las variables o constantes que use, anotándolas en la tabla que a continuación se muestra.

Regla	Especificación de variables o constantes utilizadas	Ejemplos
Formalización de proposiciones singulares		
Formalización de funciones proposicionales		
Cuantificación		
Formalización de Proposiciones cuantificadas		
Formalización de Proposiciones		

Complejas		
Proposición Universal Afirmativa (A)		
Proposición Universal Negativa (E)		
Proposición Particular Afirmativa (I)		
Proposición Particular Negativa (0)		

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Referencias

[1] S. Russell y P. Norvig, "Inteligencia Artificial. Un enfoque moderno". Prentice Hall, 1996

[2] E. Rich y K. Knight. "Inteligencia Artificial". McGraw-Hill, 1994 (lógica)

[3] N. Nilsson, "Inteligencia Artificial". Una nueva síntesis, McGraw-Hill, Madrid, 2000

[4] Mc Allister, j. "Inteligencia Artificial y Prolog". Editorial Alfa Omega/ Marcombo. México. 1999.

PRÁCTICA 3

REPRESENTACIÓN Y BÚSQUEDA HEURÍSTICA

Objetivo

- El alumno conocerá las técnicas declarativas y procedimentales para la representación y búsqueda.
- Desarrollará ejemplos de búsqueda propiamente expresados en árboles.

Introducción

Dependiendo del problema que se plantee, se ha de elegir una forma de representar el conocimiento del dominio; así que se tienen dos opciones:

- **Técnicas declarativas**: se describen los aspectos conocidos del problema, se especifica la información pero sin decir cómo usarla, describen el conocimiento del dominio como tal, se expresa con claridad y uso modular para añadir nuevos hechos, los cuales se almacenan una sola vez, también conllevan un tratamiento heurístico.

- **Técnicas procedimentales**: describe el proceso a realizar para encontrar la solución, declaran como se manipulan las entidades; son más eficientes que las anteriores, conllevan un tratamiento algorítmico por lo que son fáciles de mantener, guían las líneas de razonamiento para que la evaluación sea coherente.

Ningún sistema experto es completamente declarativo o procedimental (salvo que el problema que soluciona sea muy sencillo), ya que la especificación del conocimiento (declarativo) necesita de algoritmos para su tratamiento (procedimental). No obstante, el uso de unas u otras técnicas determinan como se representa el conocimiento.

Están técnicas son intercambiables, siempre y cuando, para las declarativas, haya un procedimiento de interpretación algorítmico.

Los métodos estarán ligados a las herramientas de representación. Para cuyo conocimiento se representa por técnicas procedimentales, los métodos son de resolución y se llevan a cabo mediante tareas de búsqueda, que son de naturaleza algorítmica (procedimental): son subtareas genéricas para la resolución de problemas.

Los métodos de resolución, que no son exclusivos entre sí, sino que, para problemas complejos, se pueden utilizar conjuntamente son:

- generar-probar
- Medios-fines
- Reducción del problema

Problemas en los que se aplican técnicas de Inteligencia Artificial

Ante un problema se pueden plantear dos situaciones:

- Que se tenga el conocimiento sobre lo que hay que hacer.
- Que haya que indagar como llegar a una solución.

Desde el punto de vista del nivel simbólico (quien lo realiza) y para los problemas para los que existe algún algoritmo que los soluciona, sus implementaciones se pueden clasificar según el costo computacional que vaya a tener, estas pueden ser:

- Problemas de tipo P; se caracteriza por tener una complejidad polinómica (dificultad a problemas de tamaño creciente. La práctica viene a decirnos que son el límite de lo "tratable".

- Problemas de tipo NP: son aquellos en los que los algoritmos que los solucionan tienen una complejidad exponencial →con tamaños pequeños del problema se consuman todos los recursos disponibles.

**La complejidad de un algoritmo es una medida de los recursos (tiempo, memoria) que requiere su ejecución en función del «tamaño» de los datos de entrada.

- Problemas con solución parcialmente conocida: en el campo del conocimiento humano, la incógnita está en cómo se formaliza el razonamiento para llegar a alguna solución, ya sea conocida totalmente o con un grado de incertidumbre.

La IA estudia precisamente como pasar del nivel de conocimiento al simbólico y conseguir un algoritmo computable para alguno de estas dos últimas categorías. Esto se hace con herramientas y métodos, que al ser independientes de dominio son:

- Aplicables a muchas clases de problemas; cada uno de ellos se pueden caracterizar encontrando unas "entidades" y "procedimientos de manipulación" comunes. Entonces, es cuando podemos crear procedimientos de resolución genéricos, independientes del dominio del problema.
- Su principal desventaja es que son poco eficientes, debido precisamente, a su carácter genérico.

Las técnicas de búsqueda son una serie de esquemas de representación del conocimiento, que mediante diversos algoritmos que permite resolver ciertos problemas desde el punto de vista de la Inteligencia Artificial.

Los elementos que integran las técnicas de búsqueda son:

- **Conjunto de estados**: todas las configuraciones posibles en el dominio.
- **Estados iniciales**: estados desde los que partimos.
- **Estados finales:** las soluciones del problema.
- **Operadores:** se aplican para pasar de un estado a otro.
- **Solucionador:** mecanismo que nos permite evolucionar de un estado a otro mediante un algoritmo aplicando los siguientes pasos:

1. Elegir el estado a explorar
2. Establecer un operador que trabaje sobre el estado elegido en el paso 1
3. Comprobar si el resultado obtenido es un estado final (es una solución del problema). Sino ir al paso 1.

Búsquedas en representación espacio de estados
Estado inicial
Estado meta
Operador
El operador cambia de un estado a otro el problema, es un procedimiento usado para modificar el estado actual del problema. Para aplicar un operador el estado actual debe satisfacer ciertas precondiciones, cada operador tiene sus propias precondiciones.
Hay un espacio de estados formado por el conjunto de los estados posibles, que existen entre el estado inicial y el estado meta, por la aplicación de los operadores.
Los *estados* del problema, definen la situación del problema y las condiciones existentes. Los *estados* son momentáneos, son resultado de las condiciones variantes del ambiente del problema. Los *estados* también pueden ser soluciones alternativas para el problema. Cada *estado* es la colección de *conocimientos* disponible en forma de estructuras simbólicas de la situación determinada del problema. El conjunto de *estados* manejados forma el espacio de *estados*, que será examinado durante el proceso de búsqueda. La meta oestado meta es el objetivo a alcanzar, la solución o respuesta final del problema.

Clasificaciones de las tareas de búsqueda

Se ha dicho anteriormente que la búsqueda es una secuencia de acciones en un orden, determinado este por la estrategia de control. Según la información utilizada para avanzar hacia una meta, las tareas de búsqueda se pueden clasificar, como:

- **Búsqueda ciega o exhaustiva**: en esta estrategia se generan estados para luego comprobar si estos cumplen con los objetivos para ser meta; si no son meta, se siguen generando otros estados. Al no tener en cuenta el conocimiento del dominio disponible (de ahí el nombre de ciega), no puede dejar ningún nodo de todos los

posibles sin examinar (de ahí el nombre de exhaustiva). Es por ello que su complejidad será la del problema exponencial en la mayor parte de los casos lo que deriva en que estos procedimientos solo sirvan para problemas pequeños.

- **Búsqueda heurística o informada:** la estrategia de control utiliza conocimiento del dominio para estimar cual es siguiente mejor estado. Dado que la búsqueda es un proceso dinámico, la estrategia de control utiliza toda la información disponible hasta ese momento. El objetivo de esta dirección informada es que el número de operadores a aplicar a un estado sea bastante menor que la exhaustiva (menos caminos inútiles), y por lo tanto mejore apreciablemente la eficiencia promedio del algoritmo (en el caso peor podría ser la misma búsqueda ciega). Hay que tener claro que las técnicas heurísticas no eliminan estados u operadores, sino que intentan mejorar el *coste* del camino a una meta.

Otra clasificación que se puede hacer es por la forma en la que se avanza en la búsqueda:

- **Búsqueda de tentativas:** se avanza en una dirección y si se llega a un punto en el que se supone que no se llega a alguna meta, se abandona este camino para retomar alguno anterior que también prometía. Esta situación se puede dar: [1] cuando a un estado no se le puede aplicar operadores y además no es meta. A esto se llamara *callejón sin salida, vía muerta o punto sin retorno.* [2] cuando las técnicas heurísticas estimen que hay un camino mejor que el que se está siguiendo.
- **Búsqueda irrevocable o sin vuelta atrás:** una vez que se ha tomado un camino, esta no se puede dejar.

La búsqueda de tentativas se puede dar tanto en exhaustivas como informadas, mientras que la irrevocable solo en las informadas. Si no se dice lo contrario, las tareas de búsqueda que se verán lo serán por tentativas.

Otra clasificación será según la naturaleza del problema. La búsqueda podrá ser:

- **Búsqueda dirigida por los datos o encadenada hacia delante.-** Consiste en seguir algún procedimiento para encontrar alguna meta. El problema se plantea como una situación inicial a partir de la cual se realiza una secuencia de acciones (aplicación de operadores) para llegar a una situación que cumpla ser una meta. También se la llama búsqueda de arriba - abajo.

- **Búsqueda dirigida por las metas o encadenada hacia atrás.-** Consiste en, dada una solución conocida, encontrar el procedimiento para llegar a esa solución. En estas se parte de una meta, a la que se le aplica algún operador que la **transforma en una o más submetas de un menor tamaño o dificultad**. Cada submeta se puede transformar en una o más submetas de manera recursiva, hasta que estas sean lo suficientemente triviales para ser solucionadas en el nivel simbólico. También se la llama búsqueda de abajo-arriba.

La mayor parte de los problemas de I.A. son planteados en términos de búsquedas dirigidas por las metas.

DESARROLLO

1. Complete la siguiente tabla.

Tipo de búsqueda	Características principales	Representación
Búsqueda ciega o exhaustiva		
Búsqueda heurística o informada		
Búsqueda de tentativas		
Búsqueda dirigida por los datos o encadenada hacia delante		

Búsqueda dirigida por las metas o encadenada hacia atrás.		

2. diseñe un árbol que represente la problemática para la adquisición de un equipo de cómputo, aplique las búsquedas dirigidas por metas y por hechos y describa en cada cado como ese recorrido.

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Bibliografía

[1] S. Russell y P. Norvig, "Inteligencia Artificial. Un enfoque moderno". Prentice Hall, 1996

[2] E. Rich y K. Knight. "Inteligencia Artificial". McGraw-Hill, 1994 (lógica)

[3] N. Nilsson, "Inteligencia Artificial". Una nueva síntesis, McGraw-Hill, Madrid, 2000

[4] Mc Allister, j. "Inteligencia Artificial y Prolog". Editorial Alfa Omega/ Marcombo. México. 1999.

PRÁCTICA 4

Formas clausales: notación semántica y cláusulas de Horn

Objetivo

- El alumno conocerá las formas clausales, su respectiva notación semántica y las cláusulas de Horn.
- Desarrollará reglas y hechos que servirán para la creación de la base de conocimiento.

Introducción

Las propiedades estructurales del sistema de la lógica de predicados de primer orden en cuanto sistema formal axiomático son las mismas que los de la lógica proposicional. En el contexto de la Inteligencia Artificial (IA) interesa recordar la decibilidad. Y ello porque es necesario resaltar una *diferencia* sustancial respecto del cálculo proposicional. Es relevante recordar que con base en la teoría de las funciones recursivas de Gödel y en la teoría de la aritmética como un todo es indecidible. La decibilidad en el cálculo de predicados bien estructurados desemboca en la creación de lenguajes de programación para la IA. El teorema es importante porque en principio puede suponer un límite inflanqueable para el desarrollo de la programación lógica en el campo de la IA. Sin embargo se debe tender en cuenta que aunque sea indecidible el sistema en el que representamos el conocimiento, sin embargo es semidecible lo cual significa que si una fórmula es un teorema del sistema a veces se pude demostrar la consistencia de esa fórmula, y si no es un teorema siempre se puede demostrar su inconsistencia a través del procedimiento de resolución que posee la propiedad de ser una refutación completa para cláusulas de Horn.

En este contexto interesa recordar el algoritmo de decisión propuesto por Herbrand: Si un conjunto de cláusulas con variables es insatisfacible o contradictorio, entonces hay un conjunto finito de sustituciones llamado universo de Herbrand del cual se puede demostrar por métodos proposicionales que es insatisfacible o contradictorio. El universo de Herbrand[1] parte de una formula en forma clausular es el conjunto de todos los términos independientes de variables que se pueden construir por la combinación de las contantes y funciones de esa fórmula. En estructura abstracta cuya utilidad estriba en cualquier fórmula de la que se ha partido para construirlo se puede interpretar en ella. Una vez que ha sido interpretada se puede aceptar que si esa fórmula es insatisfacible en este dominio también lo es en cualquier otro dominio concreto. Ahora bien, puesto que es inviable demostrar que una fórmula es insatisfacible ya que habría que analizar todas sus posibles interpretaciones, a veces infinita, se trata de encontrar en este universo abstracto una interpretación que sea insatisfacible, y considera como una condición suficiente para que la fórmula sea insatisfacible.

[1] Chang, C.L. & Char-Tung, R. "Symbolic Logic and Mechanical Theorem proving", Academic Press, Boston, 1987 p.52 definen de la siguiente manera el universe de Herbrand: "*Let H0 be the set of constants appearing in S. if no constant appears in S, then H0 is to consist of a single constant, say H0= {a}. for i=0,1,2…, let Hi+1 be the union of Hi and the set of all terms of the from f m (t1,…,tn) for all n-place functions in S, where tj,f =1,..,n, are members of the set Hi. Then Hi is called the i-level constant of S, on H°, or limit ti° Hi, is called Herbrand Universe of S*"

Un conjunto de cláusulas S es insatisfacible si y solamente si es insatisfacible un subconjunto finito de instancias de S (en el que todas las variables ligadas han recibido valor que pueda substituirla) es insatisfacible[2]. Es una regla de inferencia y procedimiento efectivo que puede servir para demostrar la validez de teoremas de la lógica de predicados, y como se ha señalado anteriormente la base en la que se sustenta e hizo posible el surgimiento del lenguaje de programación PROLOG. Ello exige transformar previamente cualquier fórmula de la lógica de predicados a su forma clausal.

Para transformar una sentencia de la Lógica de predicados en una sentencia equivalente en cláusulas de Horn hay que seguir una serie de reglas que permiten realizar esta operación y que son más complicadas, dicho proceso de transformación se puede dividir en tres grandes etapas:

1. Transformar una fórmula bien formada a la forma normal *prenex*. Se denomina forma *prenex* de una fórmula otra fórmula equivalente que consta de un prefijo en el que se hallan todos los cuantificadores sin ninguna negación y una *matriz* que no posee ningún cuantificador.

   Para generar esta clase de fórmulas se realizan los siguientes cambios:

   - Eliminar los operadores condicionales y bicondicional.
   - Reducir el alcance de la negación para que solo afecte a una formula atómica.
   - Rectificar la fórmula para que cada cuantificador afecte a una variable.
   - Agrupar todos los cuantificadores en el prefijo de la fórmula.

2. Transformar la fórmula *prenexa* a forma clausal. Supuesto que cada fórmula de la lógica de predicados es equivalente a otra en forma prenex con núcleo en forma normal conjuntiva cada una de cuyas disyunciones tiene a lo más tres miembros, está justificada lógicamente esta etapa.

Una fórmula de la Lógica de predicados es posible transformarla en dos formas canónicas denominadas conjuntiva y disyuntiva posibilitada por la interdefinibilidad de las conectivas lógicas. Se denomina forma normal conjuntiva si es una conjunción de disyunciones de literales. Se denomina literal a cualquier fórmula atómica tanto si es positiva como negativa de la fórmula las une a través del operador ***v***, es decir, que un conjunto de literales enlazados por la disyunción constituye una cláusula. La conjunción de varias cláusulas se denomina *forma clausal*. Para ello se realizan los siguientes pasos:

- Eliminar los cuantificadores existenciales de la formula *prenex* a través de funciones Skolen. Es lo que se denomina cierre existencial.
- Eliminar todos los cuantificadores.
- Transformarlo a forma normal conjuntiva.
- Distribuir la conjunción en alternativas.
- Separar los literales de la Fórmula.

[2] Rich, E., Op. Cit. P. 152; Para un análisis más exhaustivo Cfr.CUENA,J., Op. Cit. Pp. 367-401

3. Transformar la fórmula clausal a cláusulas de Horn. Una vez que ha sido reducida a forma clausal se podría aplicar el principio de resolución por refutación y por tanto demostrar la consistencia de esa fórmula. De todas maneras conviene recordar que existen diferentes estrategias que ejecutan el principio de resolución por refutación[3]. Para poder desarrollar un lenguaje eficiente de programación en el que se pudiesen representar los conocimientos y razonar sobre las mismas cláusulas de Horn que a lo ms tienen un literal positivo, esto es, a lo más una conclusión, de esto se puede deducir que existen los siguientes tipos de cláusulas de Horn:

   - **Encabezadas:** la cabeza positiva y representa la conclusión y el cuerpo contiene los literales negativos y en él se representan las condiciones.
   - **Descabezadas:** representan hechos que no dependen de ninguna condición
   - **Preguntas:** que contienen literales negativos.
   - **Clausula vacía:** denominada cláusula que termina en éxito y puede ser tomada como meta (o conclusión o hecho).

Con estas cláusulas basadas en la Lógica de Predicados se posee uno de los esquemas lógicos precisos que permiten representar hechos y relaciones entre hechos que pueden ser procesados por un sistema de aplicación o mecanismo de inferencia. Constituyen los elementos básicos del sistema PROLOG, y que se tomara como punto d referencia para juzgar la idoneidad de este tipo de representación del conocimiento.

Formas clausales: notación semántica y cláusulas de Horn

Cláusula: es una disyunción de cualquier número de fórmulas atómicas afirmadas o negadas

Las bases de conocimiento en el mundo real a menudo contiene solo clausulas, de un tipo restringido, denominadas ***cláusulas de Horn***.

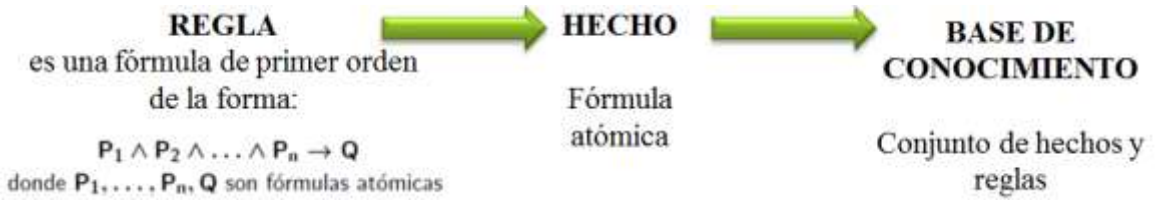
Cláusula de Horn: se caracterizan por tener uno y sólo un átomo afirmado y cualquier número de átomos negados.

P,
$Q \vee \neg P$,
$R \vee \neg P \vee \neg Q$

No todas las fórmulas se pueden transformar en cláusulas de Horn

Este tipo de cláusula, se denominan *cláusulas positivas*. El literal positivo se denomina *cabeza,* y la disyunción de literales negativos *cuerpo* de la cláusula. A este tipo de cláusula positiva que no tiene literales negativos se le denomina *hecho*. Estas forman la base de la ***PROGRAMACIÓN LÓGICA.***

[3] Farreny H.; Ghallab, M. "Elements intelligence artificialle". Hermes,Paris, 1990.
Benzaken, Cl., Systemes Formels. Masson Paris. P. 129



Ejemplos de Reglas y Hechos:

REGLAS	HECHOS
SI la luz del semáforo es verde **Y** no hay peatones cruzando **ENTONCES** continuar conduciendo	**La luz del semáforo es verde**
SI es lunes hay clase de IA **Y** son poco más de las 16:30 **Y** están presentes los alumnos de V4 leyendo esto **ENTONCES** se esta adquiriendo conocimiento de IA	**El lunes hay clase de IA**

Cláusula de Horn sin literales positivos se puede escribir como una implicación cuya conclusión es el literal *falso*. Ejemplo: (¬A∨¬B) es equivalente a (A∧B)→ *falso. A* este tipo de sentencias se les llama *restricciones de* ***integridad****;* una aplicación notoria es en las BD, donde se utilizan para indicar errores entre los datos.

La implicación o no de cláusulas de Horn, se puede realizar en un tiempo que es *lineal* respecto al tamaño de la Base de Conocimiento (BC). La inferencia lógica es un proceso que si está bien hecho, sirve de mucho en las bases de conocimiento en lógica proposicional aplicables al mundo real.

DESARROLLO

1. El alumno realizara cinco ejemplos de los tipos de cláusulas de Horn que se citaron en la introducción.

Tipo de clausula	Ejemplos
Encabezadas	
Descabezadas	
Preguntas	
Clausulas vacías	

2. Desarrolle por lo menos cinco ejemplos de Reglas y Hechos basándose en cualquier situación cotidiana o de algún tema en particular (a su libre elección).

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Bibliografía

[1] S. Russell y P. Norvig, "Inteligencia Artificial. Un enfoque moderno". Prentice Hall, 1996

[2] E. Rich y K. Knight. "Inteligencia Artificial". McGraw-Hill, 1994 (lógica)

[3] N. Nilsson, "Inteligencia Artificial". Una nueva síntesis, McGraw-Hill, Madrid, 2000

[4] Mc Allister, j. "Inteligencia Artificial y Prolog". Editorial Alfa Omega/ Marcombo. México. 1999.

PRÁCTICA 5
MECANISMOS DE INFERENCIA: ALGORITMOS DE ENCADENAMIENTO HACIA ATRÁS Y HACIA ADELANTE

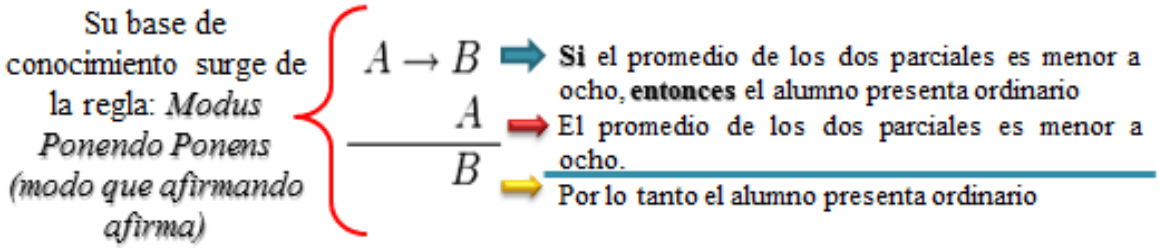
Objetivo

- El alumno conocerá el funcionamiento de los algoritmos de encadenamiento hacia adelante y hacia atrás indispensables en el funcionamiento de la base de conocimiento de reglas y hechos.
- Desarrollará reglas y hechos aplicando los tipos de encadenamiento anteriormente mencionados.

Introducción

Mecanismos de inferencia

Los dos mecanismos básicos de inferencia en sistemas de reglas, el *encadenamiento hacia a delante (forward chaining)* y el *encadenamiento hacia atrás (backward chaining)* son en esencia dos formas de aplicar la regla de inferencia *Modus Ponendo Ponens.*



El ***encadenamiento hacia adelante*** es la forma habitual de *Modus Ponendo Ponens*, pero aplicado varias veces para "encadenar" las reglas entre sí. Por ejemplo, si se sabe que se cumple $p \wedge q$ se puede concluir r y eso permite concluir a su vez s (y así sucesivamente si hubiera más reglas).

Funcionamiento de Algoritmo Encadenamiento hacia Adelante

La base de hechos (BH) **se inicializa** con los **hechos conocidos** inicialmente.

Se obtienen las consecuencias derivables de la BH:

Se comparan los hechos de la BH con la parte izquierda de las reglas;
Se seleccionan las reglas aplicables: las que tienen antecedentes conocidos (que están en la BH); las nuevas conclusiones de las reglas aplicadas se añaden a la BH (hay que decidir cómo);

Se itera hasta encontrar una condición de finalización.

Algoritmo ENCADENAMIENTO_ADELANTE(*baseConoc*, *pregunta*) {
  $deducido \leftarrow \emptyset$
  Repetir {
    Para cada regla $r = p_1 \wedge \ldots \wedge p_n \rightarrow q \in baseConoc$
      Para cada $p'_1 \wedge \ldots \wedge p'_n \in baseConoc$
        tal que $\theta \leftarrow$ UNIFICA$(p_1 \wedge \ldots \wedge p_n, p'_1 \wedge \ldots \wedge p'_n) \neq fallo$ {
        $q' \leftarrow$ SUST(θ, q)
        $baseConoc \leftarrow baseConoc \cup q'$
        $\rho \leftarrow$ UNIFICA$(q', pregunta)$
        si $\rho \neq fallo$ **devolver** ρ **fsi**
      }
  } **Hasta** $deducido = \emptyset$
  Devolver *falso*
}

Este encadenamiento, pide demostrar una pregunta a partir de una base de conocimiento *baseConoc*. El algoritmo aplica repetidamente *modus Ponendo Ponens*, añadiendo los consecuentes a la base de conocimientos. La única diferencia con la lógica proposicional es que es un poco más complejo saber cuándo se cumplen los antecedentes de una regla, ya que estos pueden aparecer variables. Para ello se utiliza el algoritmo de unificación UNIFICA. Por lo tanto, cuando se encuentran hechos de la base de conocimientos que se unifican con el antecedente de una regla, se toma la sustitución θ que realiza dicha unificación y se aplica al consecuente, para añadir el resultado a la base de conocimientos. En la práctica, cuando hay hechos que pueden activar más de una regla, se acude al mecanismo de *resolución de conflictos* que determina cuál es la que dispara en primer lugar. El algoritmo termina o bien cuando el consecuente unifica con la *pregunta*, en cuyo caso se ha realizado la demostración, o bien cuando no es posible inferir nuevos hechos. Nótese que la parte ms complicada del algoritmo es averiguar que hechos disparan qué reglas del sistema.

En el ***encadenamiento hacia atrás***, para saber si una proposición es cierta se tiene que ver si es consecuente de alguna regla y si el antecedente de esa regla se cumple. Para verificar esto último se puede aplicar el proceso de forma recursiva, encadenando las reglas hacia atrás. Para verificar por ejemplo *s* se tiene primero que verificar que se cumpla *r*, para verificar *r* se tiene que verificar $p \wedge q$. Una vez verificado esto último se ha demostrado que la proposición objetivo es cierta.

Funcionamiento:

- 1. **Se inicializa** la BH (base de hechos) con un conjunto inicial de hechos; se inicializa el conjunto de hipótesis (CH) con los objetivos a verificar; **mientras existan** hipótesis a validar en CH se **escoge una de ellas** y se valida:
  - Se **comparan** los hechos de la BH y la parte derecha de las reglas con las hipótesis;
  - Si una hipótesis está en BH eliminarla de CH; si no: buscar reglas que tengan como conclusión la hipótesis.
  - Seleccionar una, añadir las premisas no satisfechas a CH como sub-objetivos.

```
Algoritmo ENCADENAMIENTO_ATRAS(baseConoc, pregunta, θ) {
    respuesta ← ∅
    Si pregunta = ∅ Devolver ∅
    q' ← SUST(θ, PRIMERO(pregunta))
    Para cada regla r = p1 ∧ ... ∧ pn → q ∈ baseConoc {
        θ' ← UNIFICA(q, q')
        Si θ' ≠ fallo {
            pregunta' ← p1 ∧ ... ∧ pn ∧ RESTO(pregunta)
            respuesta ← ENCADENAMIENTO_ATRAS(baseConoc, pregunta', COMP(θ, θ'))
        }
    }
    Devolver respuesta
}
```

Se pide demostrar una *pregunta* a partir de las reglas en *baseConoc*. θ es la sustitución que satisface dicha pregunta, en caso de que exista (en la primera llamada al algoritmo es la sustitución nula). Se supone que la pregunta es una conjunción de átomos. Básicamente lo que se hace es tomar estos uno por uno, aplicar la sustitución θ y verificar si se unifican con el consecuente de alguna regla. Supongamos que la unificación no falla y la sustitución correspondiente es θ´. En ese caso, para demostrar la *pregunta* hay que demostrar también los antecedentes de la regla, lo cual se hace invocando al algoritmo de manera recursiva. La sustitución satisface la pregunta será la composición (función COMP) de θ con θ´. Nótese que este algoritmo es básicamente el mismo que utiliza PROLOG por lo que el algoritmo de inferencia de este lenguaje también puede verse como una implementación del encadenamiento hacia atrás.

DESARROLLO

Considere la información para influenza y gripe común que la OMS (2009) publico, a fin de contribuir en una detección oportuna y canalización adecuada.

¿Cuáles son los síntomas de la Influenza y cómo diferenciarla del catarro?

Síntomas	Catarro común	Influenza
Fiebre	Es poco frecuente en adolescentes y adultos; en los niños puede llegar hasta los 39°C	Generalmente llega a 39°C, pero puede elevarse hasta los 40°C, dura de 3 a 4 días
Dolor de cabeza	Es raro que se presente	Aparece de manera brusca y es muy intenso
Dolores musculares	Leves a moderados	Generalmente muy intesos
Cansancio y debilidad	Leves a moderados	A menudo son intensos y pueden durar de 2 a 3 semanas
Postración	Nunca	Es de inicio brusco y muy intensa
Congestión Nasal	Es frecuente	Algunas veces aparece
Estornudos	Es frecuente	Algunas veces aparece
Ardor y/o dolor de garganta	A menudo	Algunas veces
Tos	De leve a moderada	Se presenta casi siempre y puede ser muy intensa

GRUPOS POBLACIONALES DE ALTO RIESGO

CARACTERÍSTICA	GRUPOS
Edad	> 60 años
	< 5 años
Enfermedad crónica o debilitante	Cardiopatías
	Enfermedad respiratoria crónica
	Diabetes mellitus
	Cáncer
	Condiciones con depresión inmunológica
Otras condiciones	Gestación
Exposición laboral	Personal de salud
Otras exposiciones	Personas que viajan a las áreas afectadas

DATOS DE ALARMA

ADULTOS
- Dificultad para respirar
- Vómito o diarrea persistentes
- Trastornos del estado de conciencia
- Deterioro agudo de la función cardiaca
- Agravamiento de una enfermedad crónica

NIÑOS
- Fiebre alta y dificultad para respirar
- Aumento de la frecuencia respiratoria
  - Entre 2 y 11 meses: más de 50 respiraciones por minuto
  - Entre 1 y 5 años: más de 40 respiraciones por minuto
- Rechazo a la vía oral
- Trastornos del estado de conciencia
- Irritabilidad y/o convulsiones

El alumno a partir de ello desarrollara una representación gráfica de este conocimiento y aplicara mediante reglas y hechos los dos mecanismos de inferencia anteriormente expuestos.

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Bibliografía

[1] S. Russell y P. Norvig, "Inteligencia Artificial. Un enfoque moderno". Prentice Hall, 1996

[2] E. Rich y K. Knight. "Inteligencia Artificial". McGraw-Hill, 1994 (lógica)

[3] N. Nilsson, "Inteligencia Artificial". Una nueva síntesis, McGraw-Hill, Madrid, 2000

[4] Mc Allister, j. "Inteligencia Artificial y Prolog". Editorial Alfa Omega/ Marcombo. México. 1999.

PRÁCTICA 5
PROLOG: DEFINICIÓN E INSTALACIÓN

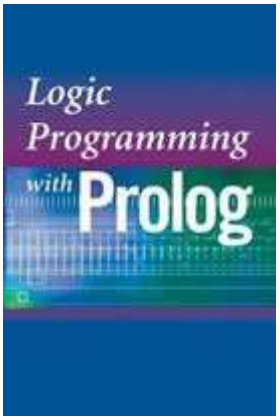
OBJETIVO

- El alumno conocerá el funcionamiento del lenguaje de programación lógica PROLOG, así como su instalación.
- Desarrollará algunos ejercicios basándose en las prácticas anteriores.

INTRODUCCIÓN

Lenguaje de Programación Lógica creado a partir de 1970 con las aportaciones teóricas de Robert Kowalski, demostraciones prácticas por Maarten van Emden y la implementación por Alain Colmerauer. Descrito, tanto la sintaxis como los operadores básicos, siguen el estándar de Edimburgo (Clocksin y Mellish[4], 1987) así como las especificaciones ISO (Covington[5], 1993 y Deransart., *et al.* 1996).

Inicialmente se pretendía usar la lógica formal como base para un lenguaje de programación, es decir, era un primer intento de diseñar un lenguaje de programación que posibilitara al programador especificar sus problemas en lógica. Lo que lo diferencia de los demás es el énfasis sobre la especificación del problema. Es un lenguaje para el procesamiento de información simbólica. PROLOG es una realización aproximada del modelo de computación de Programación Lógica sobre una máquina secuencial. Desde luego, ya que equilibra por un lado la preservación de las propiedades del modelo abstracto de Programación Lógica y por el otro lado consigue que la implementación sea eficiente.



El lenguaje PROLOG juega un importante papel dentro de la Inteligencia Artificial, y se propuso como el lenguaje nativo de las máquinas de la quinta generación ("Fifth Generation Kernel Language", FGKL) en un principio se pretendida que fueran Sistemas de Procesamiento de Conocimiento. La expansión y el uso de este lenguaje propiciaron la aparición de la normalización del lenguaje Prolog con la norma ISO.

PROLOG es un lenguaje de programación para equipos de cómputo que se basa en el lenguaje de la Lógica de Primer Orden y que se utiliza para resolver problemas en los que entran en juego objetos y relaciones entre ellos. Por ejemplo: "Jorge tiene una moto", se expresa una relación entre un objeto (Jorge) y otro objeto en particular (una moto). Más aún, estas

SWI Prolog

[4] Clocksin, W. F. y Mellish, C. S." Programación en PROLOG" Ed. Gustavo Gili, S.A., 1987
[5] Covington, Michael A."ISO Prolog. A Summary of the Draft Proposed Standard" A. I. Prog. Univ. of Georgia 1993

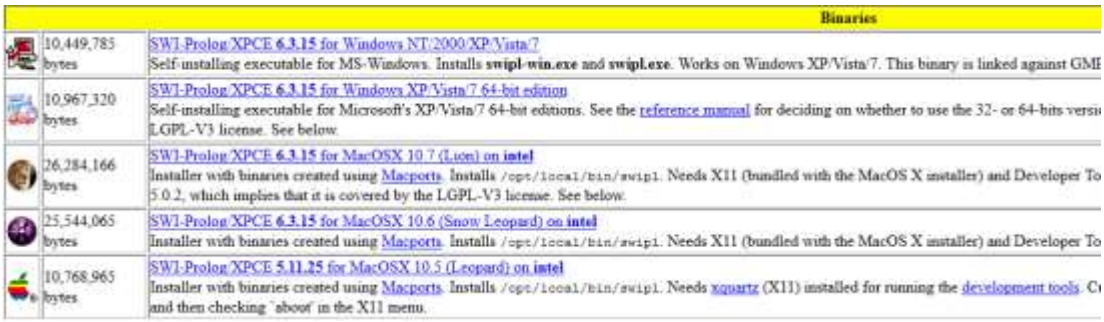
relaciones tienen un orden específico (Jorge posee la moto y no al contrario). Por otra parte, cuando se realiza una pregunta (¿Tiene Jorge una moto?) en realidad lo que se está haciendo es indagar acerca de una relación. Entre estas y otras proposiciones serán validadas en el software en posteriores prácticas.

DESARROLLO

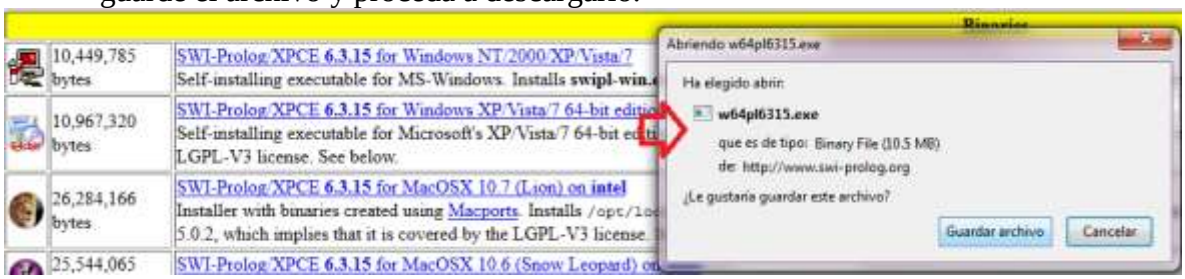
1. En cualquier buscador ubique *swi Prolog* o bien desde la página oficial www.swi-prolog.org y siga con el paso 2

2. utilice la opción de descargar:

3. seleccione la opción adecuada en función de las características de su equipo como a continuación se muestra:

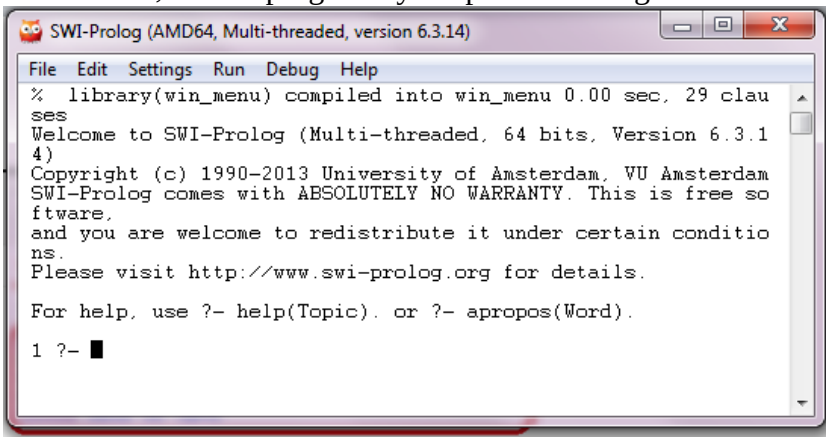


4. en este caso en particular se instaló swi-prolog para Windows XP a 64 bits, ubique la versión adecuada a su equipo, eso es muy importante para evitar confusiones, guarde el archivo y proceda a descargarlo.



5. ábralo y proceda a la descarga siguiendo las indicaciones que se le presentan

6. si todo es correcto, abra el programa y le aparecerá la siguiente ventana:



7. El alumno instalara el software en su equipo y mostrara evidencia de lo realizado.

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Bibliografía

[1] S. Russell y P. Norvig, "Inteligencia Artificial. Un enfoque moderno". Prentice Hall, 1996

[2] E. Rich y K. Knight. "Inteligencia Artificial". McGraw-Hill, 1994 (lógica)

[3] N. Nilsson, "Inteligencia Artificial". Una nueva síntesis, McGraw-Hill, Madrid, 2000

[4] Mc Allister, j. "Inteligencia Artificial y Prolog". Editorial Alfa Omega/ Marcombo. México. 1999.

PRÁCTICA 6
EXPERTLAB

OBJETIVO

- El alumno conocerá e instalara un software alternativo para declarar reglas y hechos, así como trabajar con los mecanismos de inferencia hacia adelante y hacia atrás.

INTRODUCCIÓN

ExpertLab es un motor de inferencia diseñado para la enseñanza asistida por ordenador en Inteligencia Artificial, y más concretamente, en el campo de los sistemas expertos. Sus características son: soporte de lógica proposicional, explicación durante la inferencia, ventana final de recogida de información, capacidad de respuesta a explicación (regla y camino recorrido) y encadenamiento hacia adelante y hacia atrás.

Útil para la construcción del conocimiento, siendo este el conjunto de datos de primer orden, que modelan de forma estructurada la experiencia que se tiene sobre un cierto dominio o que surgen de interpretar los datos básicos. Incluye y requiere del uso de datos e información.

Además, combina relaciones, dependencias, y la noción del saber con datos e información. Por ejemplo: *la interpretación de los valores de la química sanguínea, valores de sensores de humedad e indicar si son normales, información del series financieras, entre otros.*

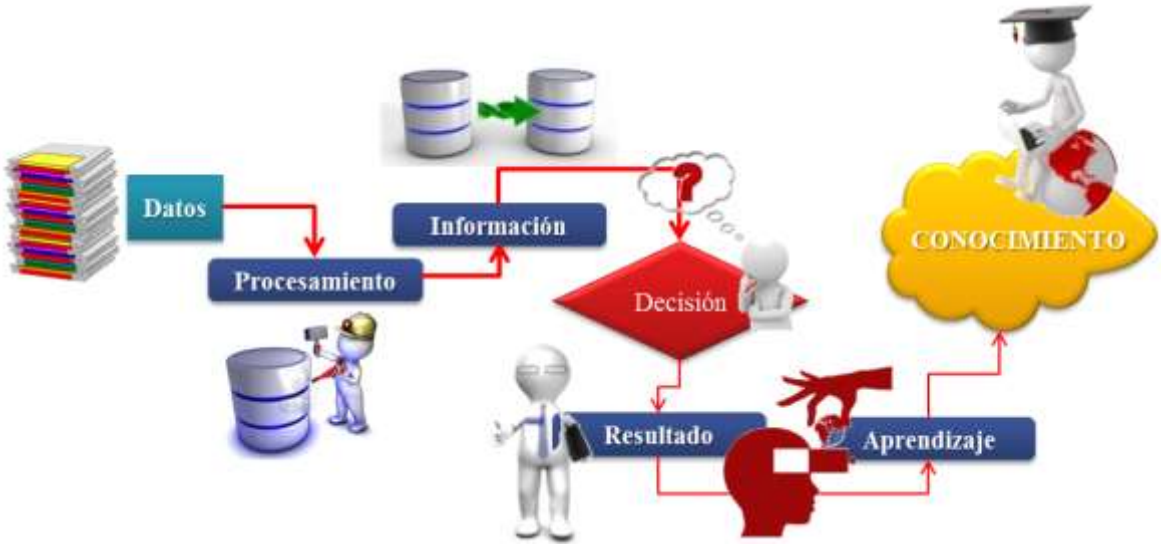
Propiedades del Conocimiento

- voluminoso
- difícil de caracterizar con precisión.
- incierto/impreciso
- cambia constantemente

Representación del Conocimiento debe ser capaz

- captar generalizaciones
- ser comprensible
- fácilmente modificable, incrementable
- ser usado en diversas situaciones y propósitos
- permitir diversos grados de detalle
- captar la incertidumbre, imprecisión
- representar distinciones importantes
- focalizar el conocimiento relevante

Una representación del conocimiento en IA es una combinación de estructuras de datos (que permiten representar mediante un formalismo determinado las "verdades" relevantes en algún dominio) asociadas con mecanismos interpretativos que permiten manipular el conocimiento representado a fin de crear soluciones a problemas nuevos.



Se manejan dos entidades:

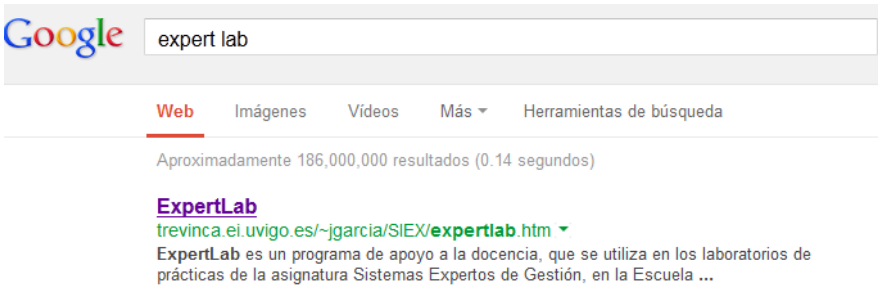
- **Hechos:** verdades en un cierto mundo, lo que se quiere representar.
- Representación de los hechos en un determinado formalismo (las entidades que se quieren manipular): **Reglas**

Clasificación de las entidades:

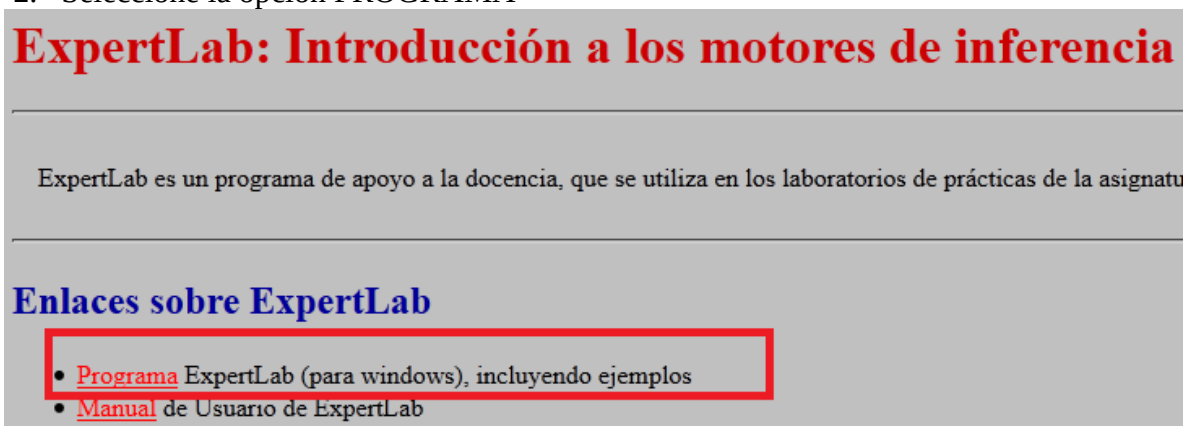
- El nivel del conocimiento, donde se describen los hechos (comportamiento y objetivos de cada agente).
- El nivel simbólico, donde se describen los objetos del nivel del conocimiento en términos de símbolos manipulables por programas.

DESARROLLO

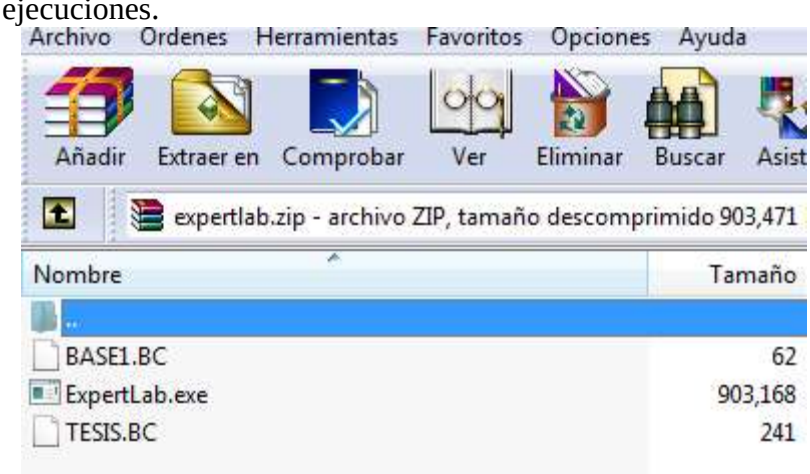
1. En cualquier buscador ubique *expertlab* o desde la dirección: **trevinca.ei.uvigo.es/~jgarcia/SIEX/expertlab.zip**



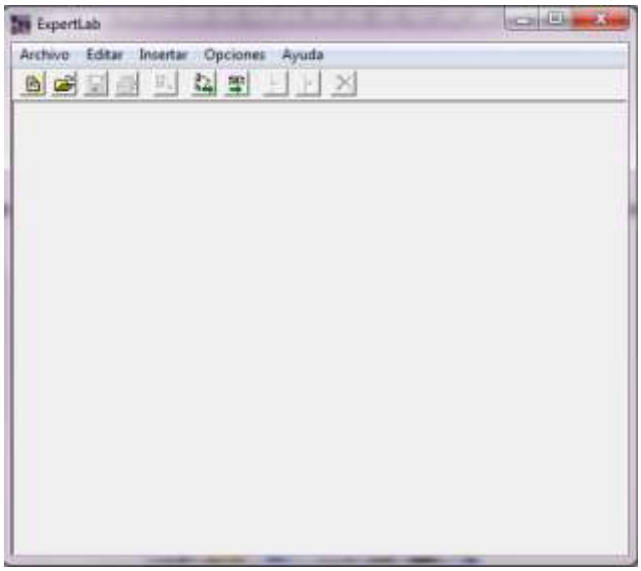
2. Seleccione la opción PROGRAMA



3. Se ejecutara lo siguiente y usted seleccionara el .exe, guárdelo en su equipo para futuras ejecuciones.



4. Si todo está bien tendrá una pantalla como esta:



5. El alumno instalara el software en su equipo y mostrara evidencia de lo realizado enviándolo a: *gutierrezcruzdo@yahoo.com.mx*

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Bibliografía

[5] S. Russell y P. Norvig, "Inteligencia Artificial. Un enfoque moderno". Prentice Hall, 1996

[6] E. Rich y K. Knight. "Inteligencia Artificial". McGraw-Hill, 1994 (lógica)

[7] N. Nilsson, "Inteligencia Artificial". Una nueva síntesis, McGraw-Hill, Madrid, 2000

[8] Mc Allister, j. "Inteligencia Artificial y Prolog". Editorial Alfa Omega/ Marcombo. México. 1999.

PRÁCTICA 7
APLICACIONES CON SWI-PROLOG

OBJETIVO

- El alumno desarrollara ejemplos con PROLOG basándose en los predicados declarados en prácticas anteriores.
-

INTRODUCCIÓN

Con base en las Prácticas anteriormente expuestas en donde se han desarrollado *predicados* para expresar propiedades de los objetos (*monádicos)* y las relaciones entre ellos (*poliádicos)* y que en conclusión se llamaran *hechos*.
La nomenclatura a considerar para la declaración de estos, básicamente se menciona a continuación:

- Nombres de todos los objetos y relaciones deben comenzar con letra minúscula.
- Primero se escribe la relación o propiedad: *predicado; ejemplo: Luis baila, declarar **baila** Luis*
- Los objetos se escriben separándolos mediante comas y encerrados entre paréntesis.
- Al final del hecho debe ir un punto (".").

Ejemplos:

```
/*ejemplos de predicado monádico*/

/*nombremujer(Person) <- Person es una mujer*/

nombremujer(yael).
nombremujer(tania).
nombremujer(alejandra).

/*nombrehombre(Person) <-Person es un hombre*/

nombrehombre(hector).
nombrehombre(jesus).
nombrehombre(juan).
nombrehombre(angel).

/*lacio(Person) <- Person tiene el pelo lacio*/

lacio(tania).
lacio(jesus).
```

```
/*ejemplos de predicado poliádico es decir RELACIONES*/

/*cursa(Person,mat)<- Person posee en este caso cursa
mat es decir materias*/

cursa(yael,calculo).
cursa(hector,archivos).
cursa(tania,ia).
cursa(jesus,basedatos).

/*caracteristicas que comparten se puede declarar como:
hace_equipo(X,Y) <- X hace equipo con Y*/

hace_equipo(yael,jesus).
hace_equipo(alejanda,tania).

/*son_amigos_de(Amigo,Amigo-a)*/
son_amigos(tania,alejandra).
son_amigos(jesus,hector).
```

Los hechos entonces se denotan:

```
notación "normal": Jesus es alumno de ISI
notación en PROLOG: alumnodeisi(jesus).
```

Las relaciones se denotan como:

```
notación "normal": Alejandra es hija de Hector
notación en PROLOG: hija(alejandra,hector).
```

Las reglas, que son sentencias condicionales se denotan como:

```
notación "normal": Si el alumno tiene promedio de los dos parciales con
ocho, entonces exenta ordinario"

notación en PROLOG: exenta(alumno):- promediodosparcialesocho(alumno)
```

Las conjunciones emplean el operador lógico AND y se utiliza coma (,):

```
notación "normal": Caro es mi tía y es hermana de mi padre

notación PROLOG: tia(X,Y):-hermana(X,Z),padre(Z,Y).
```

Regla (condición) — AND

Las disyunciones emplean el operador lógico OR y se utiliza punto y coma (;).

```
notación "normal": Puedo llegar a la escuela por la lopez o
                   la neza

notación PROLOG: escuela(X,Y):-lopez(Y,X);neza(Y,X).
```

Regla (condición) — OR

Las consultas en PROLOG utilizan la *resolución* en sus derivaciones (generalización del *modus Ponendo Ponens* junto con la *unificación*). Las consultas pueden ser interactivas mediante el indicador de comandos (**?-**).

```
calculo_diferencial(derivadas).
inteligencia_artificial(clausulas).
elctricidad_magnetismo(voltaje).

entonces la consulta en PROLOG es:

?-calculo_diferencial(derivadas).

que en normalmente se interpretaria como:
¿En calculo diferencial se ven derivadas?

la respuesta de PROLOG es: YES

otro ejemplo:


?-inteligencia_artificial(clausulas).

¿En Inteligencia Artificial se ven Clausulas?
la respuesta de PROLOG es: YES
```

También es posible utilizar variables en las consultas:

```
entonces la consulta en PROLOG es:

?-calculo_diferencial(X).

la pregunta "normal" sería: ¿Que se
ve en calculo diferencial?

la respuesta de PROLOG es:

X= derivadas

YES
```

Se llaman **cláusulas** de un predicado tanto a los hechos como a las reglas. Una colección de cláusulas forma una Base de Conocimientos.

La programación en Prolog consiste en:

- Declaración de algunos HECHOS sobre los objetos y sus relaciones,
- Definición de algunas REGLAS sobre los objetos y sus relaciones, y
- Realización de PREGUNTAS sobre los objetos y sus relaciones.

El desarrollo de un programa en PROLOG, implica un conjunto de afirmaciones (hechos y reglas) representando los conocimientos que se poseen de un determinado dominio o campo de investigación especifica. La ejecución de este programa permite la demostración de un Teorema en este Universo, es decir, la demostración de que una conclusión se deduce de las premisas (afirmaciones previas).

El sistema de PROLOG basado en un verificador de teoremas por resolución para **cláusulas de Horn**. La regla de resolución no dice que cláusulas elegir ni que literales unificar dentro de cada cláusula. La estrategia de resolución particular que se utiliza es una forma de resolución de entrada lineal (árbol de búsqueda estándar). Para la búsqueda de cláusulas alternativas para satisfacer el mismo objetivo, la primer estrategia que se ejecuta en Prolog es hacia abajo (recorrido del árbol en profundidad). Por todo esto***, el orden de las cláusulas (hechos y reglas) de un determinado procedimiento es importante, ya que determina el orden en que las soluciones serán encontradas***, e incluso puede conducir a fallos en el programa. Más importante es, si cabe, el orden de las metas a alcanzar dentro del cuerpo de una regla.

En la figura 1, se muestra a la derecha el editor de texto, que hace las veces de Base de Conocimiento; compuesta por distintos hechos y reglas que reflejan el conocimiento acerca del problema particular a formalizar. Para poder utilizarlo es necesario guardarlo en la carpeta destinada a estos programas, Posteriormente desde Prolog (ventana de la izquierda) se debe consultar el archivo deseado. Una vez consultada la base de conocimientos correspondiente, se está en condiciones de hacer las preguntas de las que se desea obtener respuesta o que resuelvan el problema. Este ciclo se repetirá continuamente (editar-guardar-consultar-preguntar) hasta que sea finalizado este proceso.

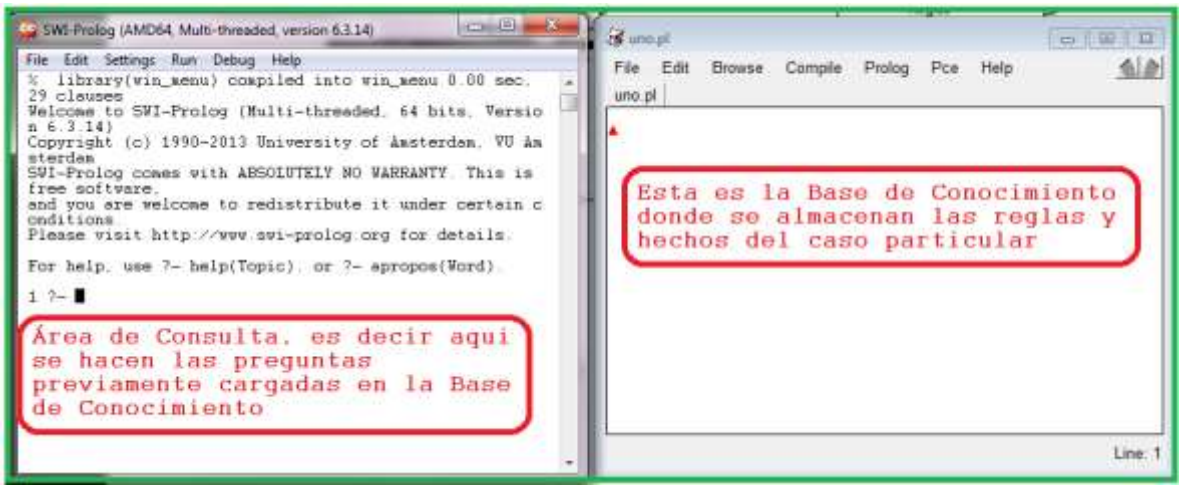


Figura 1. Ambiente de trabajo en PROLOG (consultas y base de hechos) (*Elaboración propia, 2013*).

El Lenguaje de Programación Lógica (PROLOG), considerado como un lenguaje procedimental, necesita de una metodología para construir y mantener programas largos, así como un buen estilo de programación. Sus mecanismos de control contribuyen a dar mayor potencia y expresividad al lenguaje, pero violan su naturaleza lógica. Adoptando un estilo apropiado de programación.

Metodología de Diseño "top-down"

Descomponer el problema en subproblemas es decir de lo general a lo particular o como se manejó en clase *"desmembrarlo"* y resolverlo. Para ello se solucionan primero los pequeños problemas: implementación *"bottom-up"*. Esto permite una mejor depuración («*debugged*») de los programas, ya que cada pequeña parte de programa puede ser probado inmediatamente y, si lo necesita, corregirlo.

Una vez analizado el problema y separado en partes, el siguiente paso es decidir cómo representar y manipular tanto los objetos como las relaciones del problema. Se debe elegir los nombres de los predicados, variables, constantes y estructuras de manera que aporten claridad a nuestros programas (palabras o nombres mnemónicos que estén relacionados con lo que hacen).

Posteriormente debe asegurarse de que la organización y la sintaxis del programa sean claras y legibles, mediante:

- El procedimiento al conjunto de cláusulas para un predicado dado. Agrupando por bloques los procedimientos de un mismo predicado (mismo nombre y mismo número de argumentos). Así, cada cláusula de un procedimiento comenzará en una línea nueva, y se dejara una línea en blanco entre procedimientos.

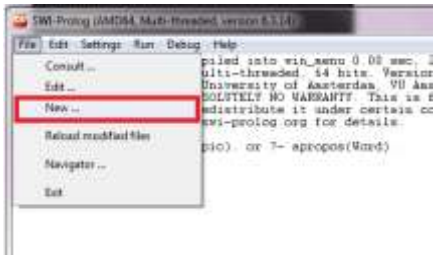
- Es recomendable añadir comentarios y utilizar espacios y líneas en blanco que hagan el programa más legible. El listado del programa debe estar "comentado". Se debe incluir para cada procedimiento y antes de las cláusulas el esquema de la relación.

- Los argumentos deben ir precedidos por el signos `+', `-' o `?'. `+' indica que el argumento es de entrada al predicado (debe estar instanciado cuando se hace la llamada), `-' denota que es de salida y `?' indica que puede ser tanto de entrada como de salida

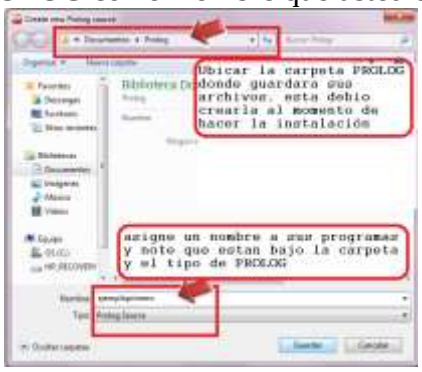
- Agrupar los términos adecuadamente. Dividir el programa en partes razonablemente autocontenidas (por ejemplo, todos los procedimientos de procesado de listas en un mismo archivo).

DESARROLLO

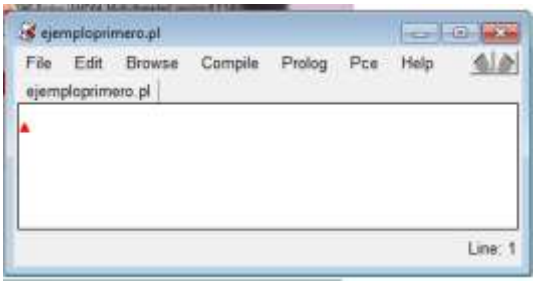
Abra un nuevo documento:



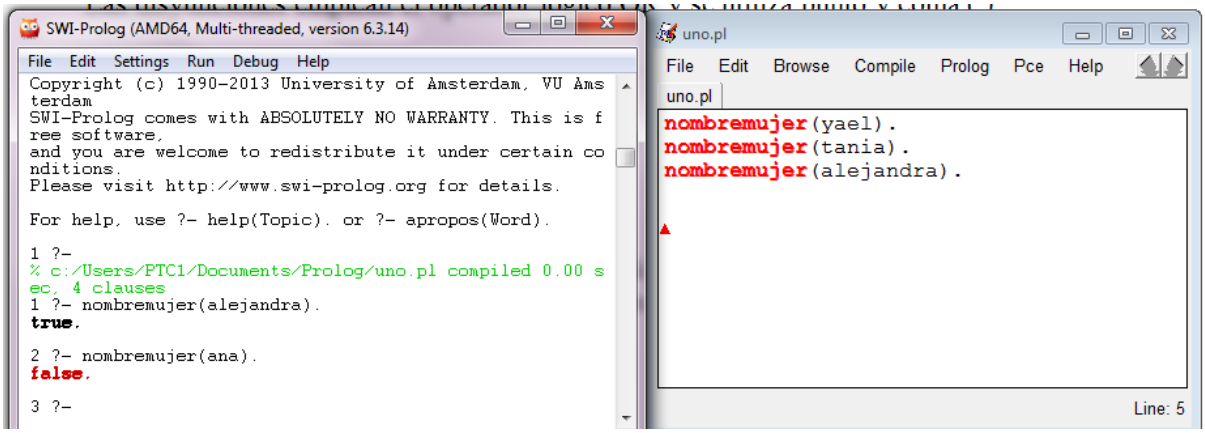
Guárdelo en la carpeta de PROLOG con el nombre que usted considere:



Aparecerá el siguiente entorno de trabajo donde se editara la información correspondiente a PROLOG:



Comience editando los ejemplos propuestos en la introducción anteriormente expuesta, como se aprecia a continuación:



El alumno ingresara en PROLOG los ejemplos que desarrollo en prácticas anteriores y mostrara evidencia de lo desarrollado enviando las respectivas ejecuciones al correo de contacto gutierrezcruzdo@yahoo.com.mx

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Bibliografía

[1] S. Russell y P. Norvig, "Inteligencia Artificial. Un enfoque moderno". Prentice Hall, 1996

[2] E. Rich y K. Knight. "Inteligencia Artificial". McGraw-Hill, 1994 (lógica)

[3] N. Nilsson, "Inteligencia Artificial". Una nueva síntesis, McGraw-Hill, Madrid, 2000

[4] Mc Allister, j. "Inteligencia Artificial y Prolog". Editorial Alfa Omega/ Marcombo. México. 1999.

PRÁCTICA 8
APLICACIONES CON EXPERTLAB

OBJETIVO

El alumno formalizara las proposiciones anteriormente formadas, pero ahora las ejecutara en EXPERTLAB y aplicara los encadenamientos hacia adelante y hacia atrás, para distinguir el funcionamiento de la base de conocimiento.

INTRODUCCIÓN

Como fue expuesto anteriormente ***ExpertLab*** es un motor de inferencia diseñado para la enseñanza asistida por computadora en Inteligencia Artificial, y más concretamente, en el campo de los sistemas expertos. Sus características son: soporte de lógica proposicional, explicación durante la inferencia, ventana final de recogida de información, capacidad de respuesta a explicación (regla y camino recorrido) y encadenamiento hacia adelante y hacia atrás. La extensión con la que se guarda este tipo de archivo es *.bc*

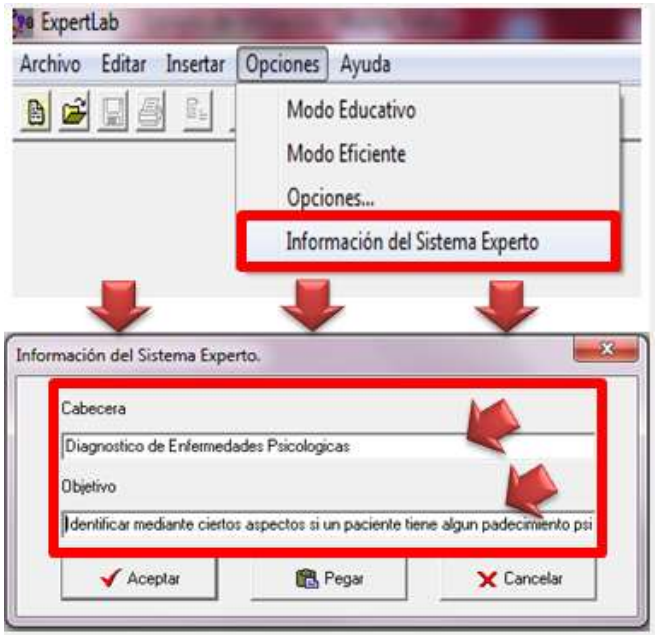
DESARROLLO

Un ejemplo para el desarrollo de reglas y hechos, y que estos a su vez sean cargados en *ExpertLab* y que sirvan de guía para el desarrollo de su propia base de conocimiento. Lea atentamente el siguiente texto, para que a partir de este se formalicen las clausulas.

La OMS define a la depresión como el más común de los trastornos mentales. Su origen es Multicasual. Los factores causales pueden dividirse en forma artificial en biológicos, genéticos y psicosociales. El cuadro clínico común se caracteriza por: Humor deprimido, Pérdida de energía (97% de los casos), Trastornos del sueño (80% de los casos), Ansiedad (90% de los casos), Trastornos del apetito, Trastornos sexuales, Síntomas cognitivos (80% de los casos) y Quejas somáticas. El hecho es que el Trastorno depresivo mayor afecta alrededor de 340 millones de personas en todo el mundo. Con una prevalencia estimada de 15 al 25%, siendo mayor en mujeres 50% de los casos antes de los 40 años, con una duración de 6 a 13 meses sin tratamiento y 3 meses con tratamiento, tiende a ser crónico el 25% recurren en los primeros 6 meses, 50% recurren en los primeros 2 años, 50-75% recurren en los primeros 5 años.

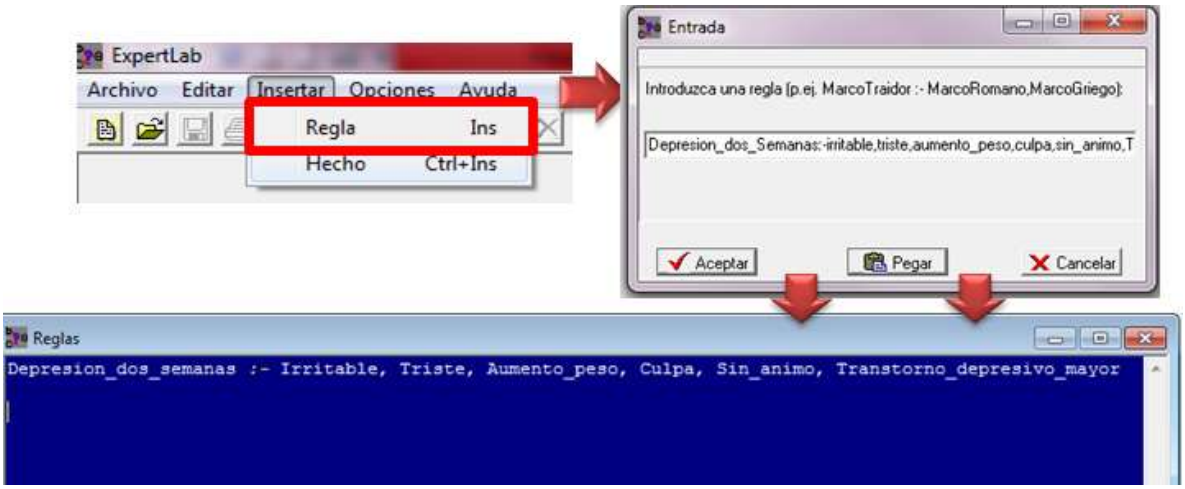
Posteriormente se procede a realizar el análisis y descomposición del mismo

1. Después de abrir el entorno de *expertlab* el alumno *etiquetara* el archivo que está realizando asignado una cabecera que hace las veces del título del trabajo y en objetivo como se indica describe la función de este sistema, siguiendo la imagen mostrada.

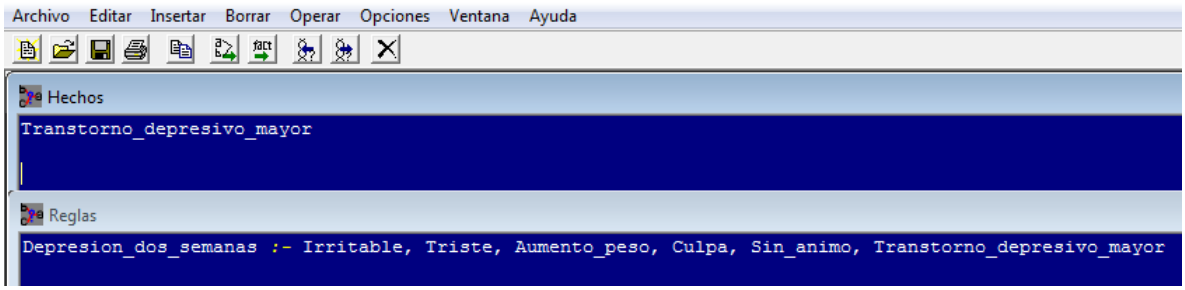


2. Para ingresar un hecho debe ir a menú→insertar→hecho y en la ventana siguiente introducir el hecho, dar clic en aceptar y el hecho estará ingresado.

3. Para ingresar una regla asociada a ese hecho debe ir a menú→insertar→regla y en la ventana siguiente introduzca la regla, dar clic en aceptar y la regla estará ingresada.



4. Observe que ahora la regla y el hecho están almacenados



El alumno ingresara en EXPERTLAB los ejemplos que desarrollo en prácticas anteriores y mostrara evidencia de lo desarrollado enviando las respectivas ejecuciones al correo de contacto gutierrezcruzdo@yahoo.com.mx

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Bibliografía

[1] S. Russell y P. Norvig, "Inteligencia Artificial. Un enfoque moderno". Prentice Hall, 1996
[2] E. Rich y K. Knight. "Inteligencia Artificial". McGraw-Hill, 1994 (lógica)
[3] N. Nilsson, "Inteligencia Artificial". Una nueva síntesis, McGraw-Hill, Madrid, 2000.

PRÁCTICA 9

SÍNTOMAS DE LOS PACIENTES, ENFERMEDADES, MEDICAMENTOS Y MÉDICOS

OBJETIVO

El alumno desarrollara al respecto a un "*SE Médico basado en síntomas, Enfermedades, Medicamentos y Médicos"*.

Generará una base de conocimientos el cual un paciente dado los síntomas que posee, pueda ser detectado que tipo(s) de enfermedad(es) pueda tener y los medicamentos que este debe tomar para su mejora así como también los médicos que pueden tratarlo.

INTRODUCCIÓN

Definición de Sistema Experto. Conocidos bajo las denominaciones de Sistemas Basado en Conocimiento (*Knowledge Based Systems, KBS*), o Sistemas Expertos S.E. (*Expert Systems*), vienen a ser programas computacionales que:

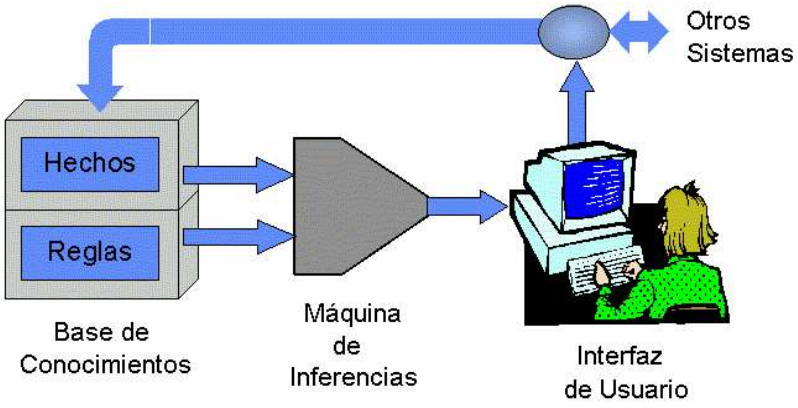
- Poseen conocimiento de un experto humano o un área de conocimiento determinada.
- Son capaces de proporcionar recomendaciones inferidas a partir de su conocimiento.
- Pueden justificar sus propias conclusiones.
- Poseen conocimiento no codificado implícitamente dentro del programa.

Dado que el término "conocimiento" resulta determinante para comprender la definición, a continuación se examina su significancia. Un especialista o experto humano aplica su conocimiento para resolver problemas, es decir, sabe cómo resolver problemas hábilmente en una ciencia o arte. Generalmente existen dos tipos de conocimientos:

a. **Declarativo:** descripciones propias del dominio del problema, i.e. hechos y asociaciones.
b. **Operativo:** se refiere a la aplicación del conocimiento declarativo dentro del proceso de resolución del problema.

Existe también la siguiente categorización del conocimiento experto:

a. **Conocimiento Empírico o Heurístico:** conocimiento declarativo y operacional que ayuda al experto a resolver problemas comunes obviando la necesidad de llevar a cabo análisis demasiado formales y detallados. Generalmente aportan una solución aprendida en base a la experiencia sin necesidad de entender cabalmente la razón por la cual funciona, i.e. heurísticas o reglas de dedo, *thumb rules*.
b. **Conocimiento formal:** incluye definiciones precisas, axiomas, leyes generales, principios y relaciones causales formales. Permite modelar dominios complejos que los humanos encuentran difíciles de comprender, permitiendo así refinar y extender el conocimiento que existe alrededor de un área o problema específico.



Arquitectura. Un sistema basado en conocimientos se compone de:

1. **Base de Conocimientos (*Knowledgebase*):** representa el conocimiento del experto y el problema en forma de hechos descriptivos y reglas de inferencia lógica. La base de conocimientos es algo más que una base de datos, ya que su mecanismo de búsqueda (*query*) es más que una simple comparación (*text matching*), de hecho (como se verá más delante), es una búsqueda donde un elemento puede "encadenar" a otro (*chaining*) utilizando comparaciones más sofisticadas (*unification and pattern matching*). Desde luego, la calidad del conocimiento de salida dependerá de la calidad del conocimiento depositado en su correspondiente base de conocimiento (*garbage-in, garbage-out*).

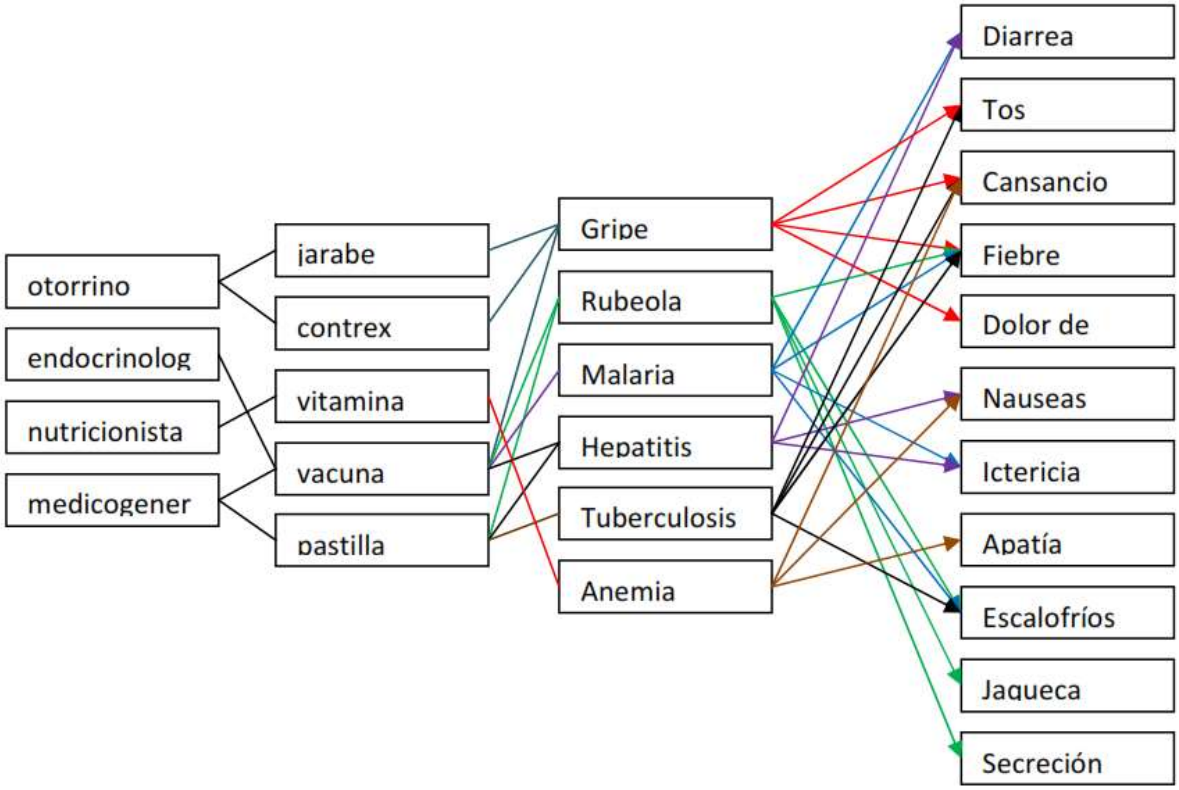
2. **Máquina de Inferencias (*Inference Machine*):** traduce reglas siguiendo sus propios algoritmos de búsqueda, control y resolución de conflictos. Dos métodos típicos de búsqueda (encadenamiento de reglas) son: 1) forward chaining y 2) backward chaining. De manera simplificada, el proceso inicia partiendo de los hechos del problema que se alimenten al sistema, e.g. un dato, lectura, señal, imágen, etc. Luego, la máquina intenta llegar a una conclusión válida buscando aquellas relgas que crea puedan cumplirse, i.e. mecanismo de encadenamiento. Cada vez que se cumple una regla, existe un nuevo hecho que de no ser la solución definitiva, puede usarse este "nuevo" conocimiento como un "nuevo" hecho en la base de conocimientos.

3. **Interfaz de Usuario (*User Interface*):** recibe y entrega información interactuando con el usuario, es decir, el usuario puede: 1) alimentar hechos, 2) proporcionar objetivos, 3) nuevas restricciones y reglas, 4) escribir programas que deban adicionarse al sistema, 5) solicitar resultados y reportes, y 6) cuestionar cómo se obtuvieron ciertas conclusiones. Dada la posible complejidad que puede existir entre el sistema y el usuario, se pueden tener distintos tipos de especialistas como usuarios, tales como: 1) el experto, 2) el ingeniero de conocimiento, 3) el programador, 4) el administrador del sistema, y 5) los usuarios finales.

DESARROLLO

Realizar un SISTEMA EXPERTO MÉDICO cuyas reglas de síntomas y enfermedades se dan a continuación:

- Declarar un conjunto de síntomas y enfermedades que existen.
- Un Paciente llega e indica los síntomas que tiene.
- El sistema experto debe reconocer que tipo de enfermedad dicha persona pueda tener además de ofrecerle medicamentos para su mejora y los doctores el cuales le pueden tratar.



Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Bibliografía

[1] S. Russell y P. Norvig, "Inteligencia Artificial. Un enfoque moderno". Prentice Hall, 1996

[2] E. Rich y K. Knight. "Inteligencia Artificial". McGraw-Hill, 1994 (lógica)

[3] N. Nilsson, "Inteligencia Artificial". Una nueva síntesis, McGraw-Hill, Madrid, 2000

[4] Mc Allister, j. "Inteligencia Artificial y Prolog". Editorial Alfa Omega/ Marcombo. México. 1999.


HUMANISMO UE TRANSF RMA
2013 2017