Universidad Nacional Autónoma de México

Posgrado en Ciencias e Ingeniería de la Computación

Notas del Curso

Conocimiento Para Compartir: Representación del Conocimiento en el Desarrollo de Sistemas Expertos

Santiago de Jesús Gonzalez Medellín

Sergio Marcellin Jacques

9 de junio de 2017

Introducción

Al observar una "nube", nos encontramos ante la posibilidad de ir mucho más allá de la realidad que observamos. El observador o usuario experto plasma en "su" sistema, las experiencias, sus vivencias, sus puntos de vista, su visión general de las cosas. Asimismo, y muchas veces de forma inconsciente, deja rastro del mundo en el que se mueve, sea éste su ámbito psicológico, laboral, doméstico, el de su barrio o el grupo de personas que frecuenta, así como su espacio histórico, geográfico y social en base a las representaciones que haga.

Las formas que emplea pueden llevarnos a confundir sus intenciones. Los escritores realistas o naturalistas buscan claramente describir el mundo que los rodea, pero existen muchas formas de llevar a cabo esta tarea. Textos que parecen referirse a un ambiente ilógico no hacen sino subrayar las contradicciones de la sociedad o de los individuos. Pensemos en las obras de teatro de Eugenio Ionesco. El absurdo de la acción encierra en realidad una crítica extremadamente ácida a la vida cotidiana del ser humano quien, arrastrado por la vorágine de la rutina, es incapaz de asumir el control de su existencia y, mucho menos, logra darle un sentido. Algo parecido podríamos decir de las obras de Franz Kafka.

El siglo XVIII abundó en cuestionamientos al orden social y político imperante, aunque no siempre de manera frontal. Jonathan Swift, por ejemplo, se lanzó a una disección despiadada de la sociedad británica de su tiempo por medio de la fantasía. Aunque son comunes las adaptaciones de sus Viajes de Gulliver como cuentos para niños, el sentido original de los liliputienses, los gigantes, las islas flotantes, los caballos parlantes, proviene de las diferentes facetas de la Inglaterra y la Irlanda que le tocó vivir. Voltaire usó viajeros interplanetarios, como Micromegas, o venidos de otros continentes, como el hurón de El ingenuo, para recalcar las fallas y las ridiculeces de su tiempo. También en Francia, Montesquieu utilizó a unos supuestos viajeros persas para efectuar una crítica semejante, y otro tanto hizo en España José Cadalso con un marroquí que habría cruzado el estrecho de Gibraltar.

Lo realizado por Montesquieu y Cadalso hace doscientos cincuenta años debe movernos a reflexionar. Lo que ellos recalcan, y esto va a ser una de las aportaciones esenciales del Siglo de las Luces, es que pueden existir múltiples puntos de vista, representaciones que sean válidos. Ya no se puede hablar de verdades absolutas y, más aún, nadie puede pretender poseer y comprender la verdad absoluta. Quizá el error más frecuente en que incurrimos es el de exigir de personas que vivieron hace siglos que vean las cosas como nosotros. La visión de un emperador romano es distinta de la nuestra por la misma razón que nuestra perspectiva no puede ser la misma que la que tendrá un australiano del siglo XLI. No es que seamos más tontos que los hombres del futuro. Simplemente, nuestro punto de vista, representación en base a nuestra percepción corresponde a la realidad que vivimos y a las experiencias, buenas y malas, de quienes nos antecedieron. O como decía Ortega y Gasset: "yo soy yo y mi circunstancia".

Aunque nos duela, nuestro punto de vista no representa ningún absoluto al que debieran aspirar todos los hombres. Esto ya nos lo explicaron Montesquieu y Cadalso. Y nosotros, cuando descubrimos modos de pensar distintos en la gente del pasado, los acusamos de ser primitivos, ignorantes, oscurantistas. Pero si pensaban de forma parecida a la nuestra, los premiamos afirmando con asombro que ya eran muy modernos. En realidad, lo que estaría sucediendo en este último caso es que nosotros somos los que seguimos pensando como los hombres del pasado. No hay que invertir las cosas. En historia, el orden de los factores sí altera el producto [Marcellin B., 2008].

En el área de la Inteligencia artificial y específicamente para la elaboración de los Sistemas Expertos (SE), cualquier modelo de representación del conocimiento incluye de una manera u otra el hecho de que un artefacto de la realidad (o "nube") esté compuesto por ejemplo, por distintos elementos, objetos y/o relaciones de diversos tipos (como otros objetos). Este "encapsulamiento"

puede verse como una estructura de un modelo de representación, como parte de varias otras estructuras, o mediante la jerarquización por medio de relaciones entre distintas estructuras que se contienen unas a otras. En este sentido, tanto las relaciones "es un" o "tiene un", como las reglas de tipo SI ....ENTONCES..... son de uso común en los trabajos de representación del conocimiento para los SE.

En los sistemas de representación del conocimiento existen diferencias en función del tipo de conocimiento con el que se esté trabajando. También es útil distinguir entre conocimiento y razonamiento, ya que el primero está ligado a los sistemas de representación y almacenamiento de la información y el segundo a los formalismos para la recuperación, conexión, inferencias y cálculos hechos con esa información (motores de inferencia), creando, muchas veces información nueva. A priori, cualquier modelo de representación del conocimiento incluye mecanismos para realizar tareas de razonamiento. *El escoger el mejor modelo para la representación del conocimiento a la hora de construir un Sistema Experto depende de la naturaleza de la aplicación y de los usuarios.*

Figura 1: •

Vivimos un nuevo estilo de vida.....

el de los marcos sin tiempo…....

.....y con poca intuición.

Sergio Marcellin, 1996.

Índice

1. Sistemas Basados en el Conocimiento (SBK) o Sistemas Expertos (SE)

En la actualidad muchas tareas son apoyadas por sistemas computacionales, los cuales se encargan de manejar, procesar y generar información de interés para aquellos que los usen. Dentro de los sistemas computacionales existen aquellos llamados *Sistemas Expertos*(SE)(o sistemas basados en conocimiento(SBK)) que hacen uso del conocimiento de un área en específico. Antes de hablar de *SBK*s es necesario hablar del medio con el que estos trabajan.

El conocimiento de un SBK consiste de hechos y heurística. Los hechos constituyen el cuerpo de la información que es compartido ampliamente, públicamente disponible y del cual existe una concordancia general entre los expertos del campo. El buen nivel de desempeño del SBK está en función del tamaño y la calidad de la base de conocimientos que esta tiene. Los SBKs que incorporan información de utilidad presentan capacidades de tomar decisiones, pueden utilizar el valor de una información para decidir si la adquieren o no y pueden calcular la sensibilidad de sus decisiones frente a pequeños cambios en las asignaciones de probabilidad y utilidad[33].

1.1. Definición del Conocimiento(K)

Para Newell [37] el conocimiento se caracteriza completamente por su funcionamiento y es aquello que media el comportamiento de un agente computacional de a cuerdo a los principios de racionalidad y además requiere de estructuras y procesos que hagan accesible este conocimiento. Por otro lado, segun la enciclopedia Sallvat [42], el conocimiento es la acción y efecto de aprender mediante la actividad. El conocimiento es un conjunto de elementos que incluye información, datos, imágenes e imaginación, actitudes, valores y "productos simbólicos", que pueden ser ciertos, aproximados o incluso falsos. Además aclara que los medios y canales de comunicación también se incluyen dentro del término conocimiento intelectual, la realidad individual y concreta y las relaciones existentes entre las cosas o conceptos.

Además de estos elementos el conocimiento toma en cuenta un factor muy importante, el "*know-how*.º el saber hacer que es la esencia de reconocer el proceso por el que se realizan acciones, cuando el conocimiento integra esta medida es cuando se convierte en un ingrediente estratégico y primario dentro de una organización.

1.2. Información vs Conocimiento

Ocasionalmente se suele confundir el concepto de información con el de conocimiento, por un lado la información viene de la relación que existe entre datos y un contexto dado dándole a los datos significado, como ya se mencionó en la sección anterior, el conocimiento incluye a la información y la aumenta con elementos más propios de un ser humano como el *saber hacer*, intuición, imaginación, valores, etc.

Para que la información derive en conocimiento se requieren prácticas que hasta hace poco tiempo se le asignaban exclusivamente a los humanos como: Comparaciones, predicciones, reconocimientos de palabras, búsqueda de conexiones semánticas, capacidades básicas de análisis, reconocimientos o esperas de satisfacer condiciones para tomas decisiones, ejecutar acciones o recuperar información, finalmente, de realizar conversaciones e intercambios de conocimientos con otros proveedores de conocimiento, prácticas que hoy también le asignamos a lo sistemas de computo llamados SBKs o SEs.

1.3. Tipos de Conocimeinto

La literatura especializada en el análisis de la información y su tratamiento como conocimiento clasifica a este último en dos grandes grupos [43].

Conocimientos teóricos: modelan el saber acerca de un tema a través de una teoría correspondiente con el problema planteado. Son "tratados" que se desarrollan a partir del análisis de los conocimientos básicos y representan una generalización de lo empírico; habitualmente se representan por estructuras simbólicas como normas de producción, modelos matemáticos, redes semánticas u objetos estructurados.

Conocimientos empíricos: son experimentales, esto es, representan el conjunto de casos prácticos observados sobre un tema (ejemplos). Son conocimientos puros que no se han tratado, analizado o modificado; representan los resultados de experiencias o los ejemplos de casos prácticos sin transformaciones[33].

1.4. Definición de SBK

Se habló por primera vez de sistemas basados en conocimiento (SBK) o sistemas expertos(SE) en una conferencia de inteligencia artificial en 1997 por Feigenbaum [18] quien más tarde diría que los SBKs son sinónimos de SE.

Un Sistema Experto (SE) es un sistema basado en computadora que integra bases de datos, memorias, mecanismos de razonamiento, agentes, algoritmos, heurísticas, para adquirir, representar, almacenar, generar y difundir conocimientos, inicialmente adquiridos a través de varios expertos humanos dentro de un dominio especifico llamado "nube"[33].

Los programas convencionales pueden ser divididos en:

- Un algoritmo, en el cual está contenido todo el conocimiento formal.
- Datos, a los cuales se les aplica el algoritmo.
- Mecanismos de entrada y salida, que contienen la información sobre la forma de comunicación con el usuario y la forma de acceder datos.

Estos programas sólo pueden dar respuestas a problemas para los cuales están específicamente programados. Si el programa necesita ser modificado para incluir nueva información, el programa entero debe ser examinado. Y para cumplir el fin para el cual fue programado, debe ejecutar una predeterminada secuencia fija de instrucciones.

Un SE independiza los procedimientos para la solución de problemas de los datos respectivos. Las modificaciones se realizan alterando la base de conocimientos sin afectar la estructura del programa completo. Estos sistemas seleccionan los medios y hechos para obtener una respuesta adecuada a una situación específica.

Con un Sistema Experto, se pueden dar recomendaciones y/o tomar acciones en las áreas de análisis, diseño, diagnóstico, planeación y control o dar solución a problemas o aplicar técnicas de enseñanza o en general recomendar, actuar y explicar las acciones que hay que tomar en actividades en las cuales normalmente, se requiere del conocimiento o saber de expertos humanos dentro de una nube especifica[33].

1.5. Tipos de SBK

En la tabla 1 se pueden observar las categorías básicas en las que se pueden categorizar a los sistemas expertos basados en funciones específicas.

Categorías	**Funciones**	**Ejemplos**
Análisis	Deducción, inferencia, diagnóstico, interpretación.	Analisis de imágenes, diagnóstico médico,desarrollo de planes.
Control	Monitoreo y supervisión.	Control de tráfico aéreo, estrategia militar.
Tratamiento de Errores	Predicción, prevensión, corrección.	Detección de fallas en equipos, reparación de autos, depuración de software
Enseñanza	Transmisión de conocimiento.	Capacitación de personal, sistemas de evaluación, sistemas educativos.
Modelado	Simulación y diseño.	Diseño de circuitos, aplicaciónes en arquitectura.

Tabla 1: Tabla de categorías de sistemas expertos

Se en la historia de los **SE** hay sistemas que marcaron la pauta para que se siguieran desarrollando este tipo de herramientas. El primer sistema considerado como Experto, fue DENDRAL, a mediados de los años sesentas, que le permitió a los ingenieros químicos orgánicos identificar posibles estructuras de moléculas orgánicas válidas, analizando su masa espectral y utilizando una base de datos químicas. Al conjuntar, la heurística de DENDRAL y el programa META-DENDRAL los usuarios podían saber cuales eran las estructuras que cumplían con las condiciones iniciales y así reducir el universo de posibles soluciones válidas y verificarlas manualmente[33].

Posteriormente a DENDRAL, señalaremos como esenciales para el área de los sistemas expertos, a partir de los años setentas, el desarrollo de los siguientes SEs, que con el tiempo, algunos de ellos, se convertirían en herramientas llamadas "shells": MYCIN, PROSPECTOR, XCON y MOLGEN[32].

2. Desarrollo de Sistemas Expertos o SBK

2.1. Metodologías para desarrollo de SBK

Básicamente para desarrollar un sistema experto se usan las mismas metodologías que con cualquier otro sistema computacional, se pueden identificar típicamente 4 etapas importantes: análisis de requerimientos, diseño del sistema, codificación y pruebas. Estas cuatro etapas pueden seguir metodologías diferentes entre ellas están las siguientes[4].

Modelo en cascada. Es un proceso secuencial, fácil de desarrollo en el que los pasos de desarrollo son vistos hacia abajo (como en una cascada de agua) a través de las fases de análisis de las necesidades, el diseño, implantación, pruebas (validación), la integración, y mantenimiento. La primera descripción formal del modelo de cascada se cita a menudo a un artículo publicado por Winston Royce W.2 en 1970, aunque Royce no utiliza el término çascada"de este artículo.

Prototipado. El prototipado permite desarrollar modelos de aplicaciones de software que permiten ver la funcionalidad básica de la misma, sin necesariamente incluir toda la lógica o características del modelo terminado. El prototipado permite al cliente evaluar en forma temprana el producto, e interactuar con los diseñadores y desarrolladores para saber si se está cumpliendo con las expectativas y las funcionalidades acordadas. Los Prototipos no poseen la funcionalidad total del sistema pero si condensa la idea principal del mismo, Paso a Paso crece su funcionalidad, y maneja un alto grado de participación del usuario.

Incremental. Provee una estrategia para controlar la complejidad y los riesgos, desarrollando una parte del producto software reservando el resto de aspectos para el futuro. Consiste en que una serie de mini-Cascadas se llevan a cabo, donde todas las fases de la cascada modelo de desarrollo se han completado para una pequeña parte de los sistemas, antes de proceder a la próxima incremental. Se definen los requisitos antes de proceder con lo evolutivo, se realiza un mini-Cascada de desarrollo de cada uno de los incrementos del sistema.

Espiral. La atención se centra en la evaluación y reducción del riesgo del proyecto dividiendo el proyecto en segmentos más pequeños y proporcionar más facilidad de cambio durante el proceso de desarrollo, así como ofrecer la oportunidad de evaluar los riesgos y con un peso de la consideración de la continuación del proyecto durante todo el ciclo de vida.

2.2. Características de un SBK

Para que un Sistema sea considerado como experto debe de cumplir con las siguientes características:

- Cuenta con habilidad para razonar por medio de diferentes métodos como son (pero no limitados a) deducción, inducción y abducción.
- Cuenta con la posibilidad de aprender, es decir, de agregar conocimiento en base a sus operaciones y que no tenía al inicio de su operación.
- Cuenta con la capacidad de explicar acciones y elecciones realizadas.
- Cuenta con uno a varios modelos para representar el conocimiento y experiencia sobre el área de domino (la nube).
- Cuenta con la capacidad de manejar conocimiento en forma dinámica que se pueden obtener de diferentes fuentes (bases de datos, sensores, agentes externos, videos, textos, redes sociales, cámaras, etc.).

- En la mayoría de los casos, debe ser capaz de manejar incertidumbre.
- Contar con los recursos de conocimientos para el SBK, es decir, que mediante el trabajo con el o los expertos del área de conocimiento se almacenen los conocimientos del experto en el sistema; adicionalmente se puedan definir indicadores (KPI's) y realizar los análisis y depuraciones necesarios para obtener los resultados que se esperan por parte del SBK. En general tener al menos en un cierto grado esta característica es deseable para cualquier sistema informático moderno.
- Debe ser capaz de producir soluciones y/o resultados satisfactorios (esto con base en el criterio del o de los expertos).
- Dado que estos sistemas, en general, son costosos es necesario verificar que proveerán un beneficio económico significativo pues de otra manera significará un riesgo económico.

2.3. Componentes básicos

Existen diversas arquitecturas de sistemas expertos, sin embargo todas ellas tienen que tener al menos los siguientes componentes.

1. **Base de datos y reglas**: Contiene el saber específico en la disciplina de la cual el sistema es experto (nube). Consiste en un conjunto de hechos (datos) y de reglas programadas. Los datos plasmados en la base juegan un papel primordial en la calidad y en las habilidades de razonamiento del sistema.
2. **Interfaz con el usuario**: Permite aceptar y reconocer un lenguaje de comandos en forma intuitiva y los traduce en instrucciones y datos para que el sistema experto trabaje.
3. **Interfaz con el experto**: Permite captar información del exterior (proporcionada por el experto) e introducirla en forma adecuada en la base de conocimiento.
4. **Máquina de razonamiento**: También llamada máquina de inferencia, este módulo, permite controlar el sistema y manejar los razonamientos similares a los realizados por los expertos de la nube.
5. **Espacio de trabajo**: Se almacenan resultados, hipótesis y decisiones intermedias, así como el estado en que el problema se encuentra en un momento dado.
6. **Modulo de aprendizaje**: Permite agregar reglas y/o procesos que no estaban definidos inicialmente dentro del sistema.
7. **Modulo explicativo**: En éste módulo se reflejan el o los porqués de las acciones del sistema. Explica sus decisiones.

2.4. Adquisición del Conocimiento

La Adquisición del K se puede entender como un proceso que incorpora dos actividades: 1) La obtención de la expertez o K del experto junto con su almacenamiento en la Base de Conocimientos y 2) la generación de K nuevo por parte del SE a partir del conocimiento adquirido del experto a lo cual se le llamará aprendizaje de la máquina.

La Adquisición del K es un proceso mediante el cual el Ingeniero del K extrae el K (hechos, reglas, procedimientos, etc.) de una fuente (libros, revistas, publicaciones, programas, etc.) así como, la información necesaria para la construcción de un SE; la analiza, la incorpora a la Base de Conocimientos y la refina.

No es fácil para el experto describir como ve los problemas por lo que un Ingeniero del Conocimiento "ideal" debe estar familiarizado con muchas áreas tales como la psicología, neurofisiología, lógica formal, estadística, así como con la terminología utilizada por los expertos del dominio que se esta estudiando. El modelo básico de la Ingeniería del Conocimiento está sustentado en el hecho de que el Ingeniero del Conocimiento debe actuar como un mediador entre el experto y la base de Conocimientos, extrayendo el conocimiento del experto, codificándolo en la Base de Conocimientos y refinándolo en colaboración con el experto hasta alcanzar una ejecución aceptable.

El experto puede realizar una serie de pasos fijos o puede seleccionar diferentes pasos en diferentes casos. Todos los casos se documentan generalmente, en forma de diagramas. ¿Cómo razona el experto a partir de los datos de entrada para obtener los datos de salida? Un experto utiliza la información inicial conocida sobre el caso a resolver y a partir de sus conocimientos genera nueva información hasta llegar a una propuesta de posible solución. Finalmente, el ingeniero debe solucionar los errores que surgen en el proceso de la representación del conocimiento y que ocurren cuando existe una diferencia entre la manera en la que el experto establece los hechos, reglas y procedimientos y la manera en la que el sistema los representa.

2.5. Representación del conocimiento

En general, los modelos de representación del K que se estudian y que se proponen dentro de las investigaciones para construir Sistemas Expertos, son para construir sistemas que puedan aprender y tomar en cuenta la semántica de los artefactos (objetos, relaciones, propiedades, estructuras y funciones) así como las incertidumbres que existen entorno al conocimiento ligado al dominio específico (la "NUBE") del SE y a su uso [33].

Generalmente en la literatura de los SE's, el término de modelos de representación es sinónimo de lenguajes de representación. Un lenguaje de representación debe poder representar en forma no ambigua, cualquier interpretación de una frase (lógicamente adecuada), debe contar con un método para traducir del lenguaje natural a esa representación y finalmente, debe permitir realizar algún tipo de razonamiento [46].

Existen varias herramientas para la representación del conocimiento en los SE's. Los modelos más utilizados son: Reglas de producción; lógica proposicional y lógica de predicados; redes semánticas (o asociativas); marcos (frames, filler and slots) así como las representaciones orientadas a objetos. El saber qué herramienta utilizar y el escoger la "mejor", depende de la naturaleza de la aplicación, de los resultados deseados y de los expertos en el campo de estudio (la nube).

Este es probablemente uno de los puntos más críticos por resolver dentro del desarrollo de un SBK. Con el K representado, podemos entonces hablar del razonamiento que se realiza dentro el SE, a través del componente llamado "maquina de razonamiento". A continuación damos una descripción de los principales modelos de representación del K utilizados en los SBKs.

3. Representación del conocimiento basada en lógicas

El conocimiento, en el principio de racionalidad, está definido completamente en términos del ambiente del agente computacional, y de las acciones que tiene que tomar para cumplir sus objetivos, entonces las soluciones son las maneras de expresar cosas sobre el ambiente. Las lógicas son los candidatos obvios [37], son un medio refinado para decir cosas sobre ambientes, se pueden encontrar muchas situaciones en las que el conocimiento del agente puede ser caracterizado por una expresión lógica de la que se puede llevar un seguimiento mecánico y detallado derivando en las acciones a tomar y ligándolas a las acciones que están sucediendo actualmente. Una lógica es solo la representación del conocimiento, no el conocimiento en sí, si se tiene un conjunto de expresiones lógicas $\{L_i\}$ de las cuales se puede decir que el agente conoce $\{L_i\}$ entonces el conocimiento que se adscribe al agente es todo lo que se puede inferir de $\{L_i\}$. Existen varios tipos de lógica, en este documento se habla de las 3 más comúnmente usadas.

3.1. Lógica proposicional

La lógica proposicional o lógica de orden cero es un sistema formal cuyos elementos más simples representan proposiciones, y cuyas constantes lógicas, llamadas conectivas lógicas, representan operaciones sobre proposiciones, capaces de formar otras proposiciones de mayor complejidad[17].

La lógica proposicional trata con sistemas lógicos que carecen de cuantificadores, o variables interpretables como entidades. En lógica proposicional si bien no hay signos para variables de tipo entidad, sí existen signos para variables proposicionales (es decir, que pueden ser interpretadas como proposiciones con un valor de verdad definido), de ahí el nombre proposicional. La lógica proposicional incluye además de variables interpretables como proposiciones simples signos para conectivas lógicas, por lo que dentro de este tipo de lógica puede analizarse la inferencia lógica de proposiciones a partir de proposiciones, pero sin tener en cuenta la estructura interna de las proposiciones más simples[23]. Una proposición es una oración la cual puede ser verdadera o falsa, por ejemplo, "*el día está soleado*" o "*mañana es lunes*". Estas proposiciones se pueden conectar mediante operadores lógicos, y esto puede derivar en otras proposiciones, los operadores lógicos básicos se muestran en la tabla 2 y sus tablas de verdad en la tabla 3.

Operador	Símbología
Conjunción	$\wedge$
Disyunción	$\vee$
Negación	$\neg$
Implicación	$\Rightarrow$

Tabla 2: Operadores Básicos

q	p	$q \wedge p$
V	V	V
V	F	F
F	V	F
F	F	F

q	p	$q \vee p$
V	V	V
V	F	V
F	V	V
F	F	F

q	$\neg q$
V	F
F	V

q	p	$q \Rightarrow p$
V	V	V
V	F	F
F	V	V
F	F	V

Tabla 3: Tablas de verdad para operadores básicos

A pesar de que el argumento sea válido, esto no quiere decir que la conclusión sea verdadera. En otras palabras, si las premisas son falsas, entonces la conclusión también podría serlo. Pero si las premisas son verdaderas, entonces la conclusión también lo es. La validez del argumento no depende del significado de las expresiones, sino de la estructura misma del argumento.

3.2. Lógica de predicados

La lógica de primer orden, también llamada lógica predicativa, lógica de predicados o cálculo de predicados, es un sistema formal diseñado para estudiar la inferencia en los lenguajes de primer orden. Los lenguajes de primer orden son, a su vez, lenguajes formales con cuantificadores que alcanzan sólo a variables de individuo, y con predicados y funciones cuyos argumentos son sólo constantes o variables de individuo. Un predicado es una expresión lingüística que puede conectarse con una o varias otras expresiones para formar una oración [12] y se pueden ver como funciones que reciben expresiones atómicas como argumentos. Los símbolos de un lenguaje de primer orden son[15]:

- Los elementos de un conjunto numerable V llamados variables $x, v, s, z, w, n, ...$ con o sin subíndices y con un orden fijo llamado alfabético.
- Los elementos de los conjuntos F_j(conjunto de funciones j-arias) y P_j(conjunto de predicados j-arios)
- Los símbolos $\neg$(negación), $\vee$(disyunción) y $\exists$(cuantificador existencial), Negación y disyunción se conocen como **conectivas**

Los términos en forma prefija, o simplemente términos, de un lenguaje de primer orden se definen como el conjunto mínimo de expresiones del lenguaje que satisfacen las dos condiciones siguientes[15]:

- Las variables son términos
- Si $a_1, a_2, ..., a_n$ son términos y f es un símbolo de función n-aria, entonces $fa_1, a_2, ..., a_n$ es un término

Por último se definen las fórmulas de un lenguaje de primer orden, que son el conjunto mínimo de expresiones del lenguaje que satisfacen las siguientes dos condiciones[15]:

- Si p es un símbolo de predicado n-ario y $a_1, a_2, ..., a_n$ son términos $pa_1, a_2, ..., a_n$ es una formula. A este tipo particular de fórmula se le llama **fórmula atómica** o átomo.
- Si A y B son fórmulas entonces también son fórmulas:
  - $\neg A$
  - $\vee AB$
  - $\exists xA$

Simbólico	Definición	Descripción
$\phi U \psi$	$(BUC)(\phi) = (\exists i : C(\phi_i) \wedge (\forall j < i : B(\phi_j)))$	Until: ψ se cumple en el estado actual o uno posterior, y ϕ se tiene que cumplir hasta esa posición. A partir de esa posición ϕ no es necesario que se siga cumpliendo.
$\phi R \psi$	$(BRC)(\phi) = (\forall i : C(\phi_i) \vee (\forall j < i : B(\phi_j)))$	Release: ϕ releases ψ si ψ se cumple hasta que la primera posición en la cual ϕ se cumple (o siempre si dicha posición no existe).
$X\phi$	$NB(\phi_i) = B(\phi_{i+1})$	Next: ϕ se cumple en el siguiente estado. (X es usado como sinónimo.)
$F\phi$	$FB(\phi) = (trueUB)(\phi)$	Finally: ϕ eventualmente se cumple (en algún lugar del camino).
$G\phi$	$GB(\phi) = \neg F \neg B(\phi)$	Globally: ϕ se tiene que cumplir en todo el camino.
$A\phi$	$AB(\psi) = (\forall \phi : \phi_0 = \psi \rightarrow B(\phi))$	All: ϕ se tiene que cumplir en todos los caminos empezando en el estado actual.
$E\phi$	$EB(\psi) = (\exists \phi : \phi_0 = \psi \wedge B(\phi))$	Exists: existe al menos un camino que parte en el estado actual en el cual ϕ se cumple.

Tabla 4: Tabla de operadores temporales [2]

3.3. Lógica temporal

La lógica temporal es una extensión de la lógica clásica para permitir la formalización de enunciados que incluyan precisiones acerca del momento del tiempo en que han tenido lugar. En lógica clásica de proposiciones dos enunciados como "está lloviendo" y "lloverá" deben ser formalizados o bien como dos proposiciones completamente diferentes o como la misma proposición, la lógica temporal nos permite formalizarla como la misma acción en dos momentos diferentes del tiempo, nos permite discriminar si un hecho tiene lugar en el presente, en el pasado o en el futuro. La lógica temporal es utilizada en filosofía con el objetivo fundamental de analizar y clarificar conceptos clave recurrentes en su historia; la mayor parte de ellos señalados por Aristóteles como la casualidad y la necesidad, entre otros. No hay una sola lógica temporal sino que existen muchas dependiendo de la concepción del tiempo que tengamos o que deseemos utilizar[3]. En lógica temporal aparecen los mismos operadores que en una lógica de primer orden, junto con otros nuevos, entre los que se pueden encontrar: Siempre, algunas veces y nunca. En la tabla 4

Las reglas de deducción en todos los sistemas son Modus Ponens y Generalización de L (cuando aparezca) de H y de G: si $\_\alpha$, entonces $\_L_\alpha$, $\_H_\alpha$ y $\_G_\alpha$.

Ax_0. Todas las tautologías de la lógica clásica de proposiciones

Ax_1. $G(A \rightarrow B) \rightarrow (GA \rightarrow GB)$

Ax_2. $H(A \rightarrow B) \rightarrow (HA \rightarrow HB)$

Ax_3. $A \rightarrow HFA$

Ax_4. $A \rightarrow GPA$

Ax_5. $FFA \rightarrow FA$ (transitividad)

Ax_6. $PPA \rightarrow PA$ (transitividad)

Ax_7. $PFA \rightarrow (PAvAvFA)$(linealidad hacia delante)

Ax_8. $FPA \rightarrow (PAvAvFA)$(linealidad hacia atrás)

Ax_9. $GA \rightarrow FA$(futuro infinito)

Ax_{10}. $HA \rightarrow PA$(pasado infinito)

Ax_{11}. $FA \rightarrow FFA$(densidad)

Los axiomas 0, 1, 2, 3 y 4 constituyen un sistema mínimo. Este se denomina K_t y fue desarrollado por Lemmon en 1965. Añadiendo al sistema mínimo los axiomas 5 y 6 se obtienen modelos transitivos. Si, además, se le añaden los axiomas 7 y 8 se obtiene un sistema para el tiempo lineal. A cualquiera de estos sistemas añadiéndoles los axiomas 9 y 10 se obtiene un sistema de tiempo infinito, y con el axioma 11 un sistema para el tiempo denso.

4. Representación del conocimiento basada en Reglas de producción

En los SE y otros sistemas basados en conocimiento, el método más utilizado y popular para representar el conocimiento es a través de reglas de producción. Su simplicidad y similitud con el razonamiento humano, han contribuido para su popularidad en diferentes dominios. Las reglas son, entonces, un importante paradigma de representación del conocimiento. Los sistemas que utilizan este método de Representación del conocimiento se les conoce como Sistemas Basados en Reglas de Producción (SBRP). Fueron usados primero en lógica simbólica por Post (1943) quien comprobó la importancia de representar mediante un sistema de reglas de producción, cualquier sistema de matemáticas o lógica. También en la lingüística por Chomsky (1957) se utilizan reglas de reestructuración en el reconocimiento sintáctico de frases de lenguaje natural. Newell y Simon 70's proponen los sistemas de producción como un modelo psicológico del comportamiento humano, en este modelo, parte del conocimiento humano se representa en forma de producciones o reglas de producción y se asemeja al proceso de memoria humano: memoria a corto plazo (deducciones intermedias) y memoria a largo plazo (producciones). Normalmente las reglas de producción se ven como un formalismo en el cual representar el conocimiento y es el formalismo más usado en los sistemas expertos. Originalmente las producciones eran reglas gramaticales para manipular cadenas de símbolos.

4.1. Fundamentos

Los interpretadores de reglas generalmente ejecutan un algoritmo encadenado hacia adelante para la selección de producciones para ejecutar para llegar a los objetivos actuales, los cuales pueden incluir la actualización de la información del sistema o creencias. La porción de condición de cada regla (del lado de mano izquierda o LHS) es puesto a prueba en contra de él estado de la corriente de la memoria de trabajo. Sistemas expertos en tiempo real, en contraste, frecuentemente tienen que escoger entre producciones exclusivas mutuamente, ya que las acciones toman tiempo, solo una acción puede ser llevada a cabo o (en el caso de un experto en sistemas) recomendado. En tales sistemas, el interpretador de normas o máquina de interferencia, funciona a través de dos pasos: relacionando normas de producción en contra de la base de datos, seguido por una selección de cuáles de las normas que concuerdan son aplicadas y ejecutadas a las acciones seleccionadas. Relacionar normas de producción en contra de la memoria de trabajo. Los sistemas de producción pueden variar el poder expresivo de las condiciones de producción de las normas. De acuerdo a el patrón algoritmo relacionado el cual colecta normas de producción con condiciones coincidentes puede tener un rango de NAIVE, tratando todas las reglas en secuencia, parando a la primera coincidencia a el optimizado, en el cual las reglas son compiladas a condiciones de una red interrelacionada [11]. Una de las características sobresalientes de los SBRP es su potencialidad de aprendizaje, es decir, que a partir de una Base de conocimiento inicial, el sistema es capaz de generar o simplificar las reglas de inferencia que rigen su comportamiento, esto último no lo hace en automático. Un SBRP está compuesto por reglas de producción que representan el conocimiento utilizando un par ordenado (A, B) que puede representarse en el lenguaje de la lógica proposicional como A entonces B. Una regla de producción es una estructura del tipo:

SI <antecedente> **ENTONCES** <consecuente>

En donde el antecedente (parte izquierda = SI o IF) es el conjunto de condiciones, premisas o situaciones y el consecuente (parte derecha = ENTONCES o THEN) contiene la conclusión, acción o respuesta resultante si las premisas son satisfechas, como:

Si la luz es roja ENTONCES deténgase

Si existe el hecho de que la luz sea roja, esto concuerda con el patrón "la luz es roja", de modo que la regla se satisfizo y se ejecuta la acción "deténgase".

En resumidas cuentas podemos decir que un SBRP el proceso de inferencia opera en dos fases: [25]

- Fase de reconocimiento. En esta fase se llevan a cabo las siguientes actividades:

  Selección de reglas pertinentes. Depende de la situación en curso de tratamiento, del tipo de encadenamiento (hacia adelante o hacia atrás)

  Resolver el conflicto de resolución cuando existe más de una regla aplicable a través de la aplicación de criterios como:

  Establecimiento del orden en los datos.

  Clasificación de las reglas por prioridad de ejecución.

  Ejecución de la regla instanciada más recientemente.

  Aplicación metareglas.

- Fase de de acción. Es esta fase se ejecutan las acciones establecidas por lasreglas durante el proceso de reconocimiento.

4.2. Control, resolución de conflicto: Meta-reglas

Las meta reglas son aquellas que establecen restricciones sobre las reglas de producción, indican como se tienen que manejar en ciertas situaciones y llevan un control sobre el flujo de las mismas. Las reglas pueden ser evaluadas por encadenamiento hacia atrás o hacia adelante

Encadenamiento hacía atrás. Para determinar si una decisión debe de ser tomada, se trabaja hacia atrás buscando justificaciones para la decisión que se tomó. Eventualmente una decisión debe de estar justificada por hechos. **Encadenamiento hacia adelante**. Dados algunos hechos, trabajar hacia adelante en la red de inferencia, descubriendo las conclusiones que se pueden derivar de los datos. Hasta que un problema sea resuelto o no haya reglas cuya parte del "if" sea satisfecha por la situación actual

1. Recolectar reglas cuyas partes "if" son satisfechas
2. Si más de una regla es satisfecha en su precedente, usar una estrategia de resolución de conflictos para eliminar todas excepto una.
3. Ejecutar la parte "then" de la regla.

Cuando se encuentran dos o más reglas que se cumplen por el estado actual y existe un conflicto entre ellas se debe de implementar una estrategia de resolución. Hay 4 variantes de estrategias de resolución

1. Ordenar las reglas por orden de uso y escoger la usada más recientemente.
2. Ordenar las reglas por orden de uso y escoger la usada menos recientemente.
3. Ordenar las reglas elegibles de acuerdo al uso de los datos, escogiendo aquella que haya hecho referencia al dato usado más recientemente.
4. Ordenar las reglas elegibles de acuerdo al uso de los datos, escogiendo aquella que haya hecho referencia al dato usado menos recientemente.

4.3. Tratamiento de incertidumbre

El siguiente paso en la aplicación de SBRP lo encontramos en los algoritmos de Markov concebidos como una estructura de control para los sistemas de producción usados como base de un SE pero poco eficaz para un sistema con muchas reglas (debido a que si no se satisfacen las reglas de mayor prioridad se efectúa una búsqueda en las reglas de menor prioridad). Con la finalidad de mejorar la eficiencia en la solución problemas resultantes del emparejamiento de reglas en el Motor de inferencias, Charles L. Forgy desarrolló el algoritmo Rete en 1979, que es un rápido igualador de patrones, para el proceso de apareamiento ("match"), obtiene su velocidad del almacenamiento de información sobre las reglas de una red, es decir, compara los hechos con los patrones de reglas y determina cuales de ellas han satisfecho sus condiciones. Las relaciones entre las reglas y los hechos son formadas durante la ejecución, basándose en los patrones que se identifican con los hechos. Al contrario de cualquier ciclo-acto de reconocimiento, en el cual se tienen que igualar los hechos con todas las reglas en cualquier ciclo-acto reconocimiento, el algoritmo Rete sólo busca los cambios en las correspondencias de cada ciclo, esto acelera en gran medida la correspondencia de los hechos con los antecedentes pues los datos estáticos que no cambiaron de un ciclo a otro pueden pasarse por alto. Posee características específicas como:

- El disparo de una regla produce, generalmente, pocos cambios en la Memoria de trabajo.
- Un mismo patrón suele utilizarse en varias reglas.
- Construye y mantiene el grafo Rete enraizado, dirigido y acíclico:

  Nodos: Representan patrones de hechos (menos la raíz)

  Caminos: Representan las condiciones de una regla (desde la raíz)
- Cada nodo contiene información acerca de los hechos que emparejan con los patrones de los nodos desde la raíz, junto con las asociaciones necesarias de las distintas variables.

4.4. Ejemplos

Los *SBK* implementados en la industria y el comercio han traído grandes beneficios incluyendo:

- incrementos en el orden de magnitud de la velocidad de cumplimiento de tareas complejas
- incremento en la calidad
- reducción de errores
- reducción en los costos
- reducción en el personal requerido
- reducción en tiempos de entrenamiento
- retención de conocimiento volátil o portable
- mejora en la toma de decisiones
- mejora en servicio a clientes

Gracias a el potencial de los *SBK* se han podido implementar en muchas áreas brindando apoyo a los usuarios, en la figura 2 se muestra la distribución del uso de SBK [24].

Otros ejemplos de *SBK*s que se pueden encontrar son[1]:

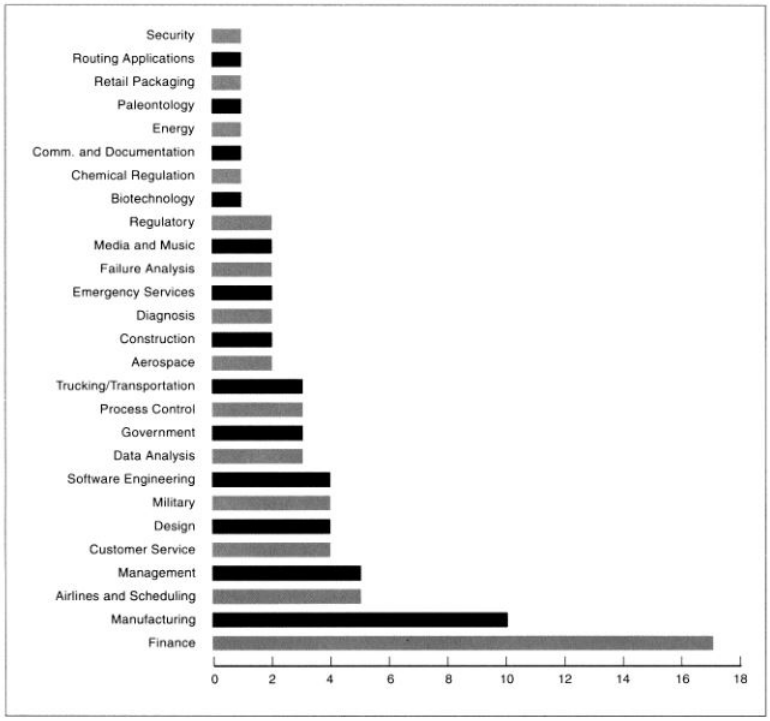

Figura 2: Distribución de uso de SBK[24]

- **DENDRAL**. Es capaz de calcular o descubrir hechos relativos a las estructuras moleculares a partir de unos datos químicos sin elaborar.
- **MYCIN**. El mas famoso de todos, diagnostica infecciones en la sangre y meningitis y además sugiere el tratamiento que se debe seguir en cada caso.
- **PUFF**. El hermano menor de MYCIN, que diagnostica y trata enfermedades del pulmón.
- **MOLGEN**. Ayuda a los biólogos que trabajan en el campo del ADN y la ingeniería genética.
- **XCON**. Ayuda a la configuración automática y diseño de equipos de cómputo.
- **PROSPECTOR**. Apoya a los geólogos en la explotación de minas.
- **PROGRAMMERS APPRENTICE**. Se trata de un sistema que ayuda a la escritura de programas.
- **EURISKO**. Sistema experto capaz de aprender a medida que funciona, que crea circuitos microeléctricos tridimensionales
- **GENESIS**. Permite a los científicos planificar y simular experimentos en el campo de la unión de genes
- **TWIRL**. Simulaciones de guerras completas y guia de mejores acciones posibles a realizar, en casi todas las situaciones.
- **RI**. Programa utilizado para el descubrimiento de yacimientos petrolíferos bajo aguas marinas.

5. Representación del conocimiento basada en Redes Semánticas, grafos y árboles

5.1. Redes semánticas

Son herramientas gráficas que nos permiten representar conocimiento lingüístico mediante objetos y relaciones, su concepción se basa en la asociación de conocimientos que realiza la memoria humana. Actualmente, se utiliza el término redes asociativas (una forma más amplia) ya que no sólo se usa para representar relaciones semánticas, sino también para representar asociaciones físicas o causales entre varios conceptos u objetos.

Los objetos se denominan también nodos, los nodos son usados para representar elementos del dominio, un atributo, un estado, una entidad o un evento (elementos del conocimiento) y las relaciones entre nodos se denominan asociaciones, arcos o enlaces, los arcos simbolizan las relaciones entre los elementos, la forma gráfica de ubicar un enlace es como un vector desde un nodo a otro, titulado con el nombre de las relaciones representadas. Cada nodo y cada asociación en una red semántica, deben estar asociados con objetos descriptivos.

El razonamiento con redes semánticas es directo puesto que las asociaciones se pueden hacer simplemente rastreando los enlaces en el sistema, a este mecanismo se le llama propagación de la activación. Sin embargo, ninguna regla semántica rigurosa guía tal razonamiento. La interpretación de las estructuras de la red depende solamente del programa que las manipula, es decir, que no existe ninguna convención del significado, por esta razón, las inferencias que se derivan de la manipulación de la red no son necesariamente válidas. La inferencia mediante el aparejamiento es otra técnica de razonamiento usada en redes semánticas, se basa en la construcción de una fracción de red que es una mezcla de nodos con valores definidos y nodos cuyos valores se requieren pero son desconocidos.

Las redes asociativas tienen dos ventajas sobre los sistemas basados en reglas y sobre los basados en lógica:

1. Permiten la declaración de importantes asociaciones, en forma explícita y sucinta.

2. Debido a que los nodos relacionados están directamente conectados, y no se expresan las relaciones en una gran base de datos, el tiempo que toma el proceso de búsqueda por hechos particulares puede ser significativamente reducido.

Entre las desventajas de las redes asociativas, se pueden mencionar:

1. No existe una interpretación normalizada para el conocimiento expresado por la red. La interpretación de la red depende exclusivamente de los programas que manipulan la misma.

2. La dificultad de interpretación a menudo puede derivar en inferencias inválidas del conocimiento contenido en la red.

3. La exploración de una red asociativa puede derivar en una explosión combinatoria del número de relaciones que deben ser examinadas para comprobar una relación.

5.2. Modelos Gráficos

Modelos gráficos probabilisíticos usa una representación basada en grafos como base para de forma compacta representar un espacio probabilístico multidimensional. En la estructura de grafo los vértices corresponden a las variables del dominio mientras que los arcos entre vértices corresponden a las interacciones probabilísticas entre éstos. Este es la primera ventaja de modelos gráficos, su representación es transparente gracias a la estructura de grafo. Por ejemplo, un experto en el dominio

sobre el que se aplique modelos gráficos podría entender las dependencias entre variables aleatorias sólamente observando el grafo, sin ser distraído por los datos que subyacen. La segunda ventaja es la batería de algoritmos que modelos gráficos ofrece para efectuar inferencia en el modelo. La tercera ventaja es el conjunto de algoritmos para aprender el modelo a partir de un conjunto de aprendizaje. Estas tres ventajas son las esenciales para el uso de modelos gráficos. La red bayesiana es la más popular (un caso particular muy conocido es la Naive Bayes que es usada como clasificador, por ejemplo, de spam)[26].

Las redes bayesianas es un tipo de modelo del framework PGM usadas en expresión genética (análisis genético), medicina, entre otras disciplinas. En particular, como se ha comentado anteriormente, el modelo Naive Bayes, que es una simplificación de una red bayesiana, es usado como método estándar de clasificación del cual existen implementaciones en múltiples librerías de programación. En una red bayesiana el grafo es dirigido acíclico, los vértices son variables aleatorias y los arcos corresponden intuitivamente a influencias directas entre variables.

Definición 1. *Una estructura de red bayesiana G es un grafo dirigido acíclico donde los representan variables aleatorias $X_1, ..., X_n$. Sea $Pa_{X_i}^G$ la expresión de los padres de X_i en G, y $NonDescendants_{X_i}$ la expresión de los vértices no descendientes de X_i en G. Entonces G contiene el siguiente conjunto de asumpciones de independencia condicional, llamadas independencias locales, y denotadas por $I_l(G)$:*

$$\text{Por cada variable } X_i : \left(X_i \perp NonDescendants_{X_i} \mid Pa_{X_i}^G\right)$$

El concepto de $I_l(G)$ se generaliza mediante el concepto I-map que flexibiliza la compatibilidad entre distribuciones de probabilidad y estructuras de red bayesiana. No obstante, no es imprescindible para la definición de red bayesiana. Pero antes de definir la BN primero hay que ver la compatibilidad entre la estructura y una distribución:

Definición 2. *Sea G una estructura de BN sobre las variables $X_1, ..., X_n$ y P una distribución de probabilidad sobre el mismo espacio que define G, entonces se dice que P factoriza en G si P puede ser expresada como el siguiente producto:*

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P\left(X_i \mid Pa_{X_i}^G\right)$$

Con esto se puede definir el concepto de red bayesiana:

Definición 3. *Una red bayesiana es el par $B = (G, P)$ donde G es una estructura de red bayesiana y P es una distribución de probabilidad tal que factoriza en G y es especificada mediante las distribuciones de probabilidad condicionadas asociadas con los vértices de G.*

5.3. Árboles

Un árbol es un tipo especial de gráficas con la propiedad de que entre cualesquiera 2 pares de nodos hay un único camino y no contiene ningún ciclo. Los árboles de decisión son un enfoque muy popular y comúnmente usado para realizar tareas de clasificación. Estos clasificadores comienzan con el conjunto de entrenamiento y sus etiquetas de clase asociadas, la raiz es la característica principal, cada nodo interno representa atributos de prueba, cada rama representa la salida de la prueba y cada nodo hoja representa las etiquetas de clase. Para identificar la etiqueta de prueba de una muestra desconocida el árbol de decisión traza un camino desde la raiz hasta el nodo hoja que contiene a la clase de esa muestra. Tres de las implementaciones más típicas de este modelo son clasificadores ID3, clasificadores C4.5 y "Random Forests"[44].

Clasificador ID3. Es el algoritmo básico para árboles de decisión. Este algoritmo construye un árbol de decisión a partir de los datos(discretos en naturaleza). Para cada nodo, selecciona el

mejor atributo. Y su mejor atributo es seleccionado usando el criterio de selección Ganancia de Información[16], que indica que tan informativo es un nodo, y el atributo con la mayor cantidad de ganancia de información es seleccionado como nodo de separación.

Algoritmo C4.5. Este algoritmo es desarrollado en base al ID3[40]. ID3 también se usa para generar árboles de decisión sin embargo no garantiza una solución óptima cuando se trata de analizar datos continuos, en cambio el algoritmo C4.5 supera ese inconveniente.

"Random Forests". Opera construyendo múltiples árboles de decisión[29]. Este algoritmo está basado en "bagging"[19], después de generar múltiples subconjuntos de entrenamiento aleatoriamente, el algoritmo construye un árbol por cada uno, esta técnica es llamada selección de separación aleatoria y los árboles se conocen como árboles aleatorios.

5.4. Redes de Petri

Una Red de Petri es una representación matemática o gráfica de un sistema a eventos discretos en el cual se puede describir la topología de un sistema distribuido, paralelo o concurrente. La red de Petri esencial fue definida en la década de los años 1960 por Carl Adam Petri[38]. Son una generalización de la teoría de autómatas que permite expresar un sistema a eventos concurrentes.

Mediante una red de Petri puede modelarse un sistema de evolución en paralelo o eventos concurrentes compuesto de varios procesos que cooperan para la realización de un objetivo común. La presencia de marcas se interpreta habitualmente como presencia de recursos. El franqueo de una transición (la acción a ejecutar) se realiza cuando se cumplen unas determinadas precondiciones, indicadas por las marcas en las fichas (hay una cantidad suficiente de recursos), y la transición (ejecución de la acción) genera unas postcondiciones que modifican las marcas de otras fichas (se liberan los recursos) y así se permite el franqueo de transiciones posteriores.

Definición: Una red de Petri es un conjunto formado por $R = \{P, T, Pre, Post\}$, donde P es un conjunto de fichas de cardinal n, T un conjunto de transiciones de cardinal m, Pre la aplicación de incidencia previa que viene definida como

$$Pre : P \times T \to N$$

y $Post$ la aplicación de incidencia posterior que viene definida como

$$Post : P \times T \to N$$

Definición: Una red marcada es un conjunto formado por donde R es una Red de Petri como la definida, M es una aplicación denominada marcado y

$$M : P \to N$$

Se asocia a cada marca un número natural por lo tanto en donde el número de marcas es descrita por la cardinalidad del conjunto de marcas en la red.

6. Representaciones basadas en Marcos y en Objetos

6.1. Marcos ("Frames")

En 1975 Marvin Minsky sugirió la idea de utilizar grupos de procedimientos orientados a objeto para reconocer y hacer frente a nuevas situaciones[35]. Minsky uso el término marco como la estructura de datos utilizada para representar a estas situaciones. Aunque la intención original de la aplicación de los marcos como una representación del conocimiento fue el reconocimiento, la idea de agrupar los procedimientos relacionados tuvo una mucho más amplia aplicabilidad en el razonamiento.

Los lenguajes de marcos proporcionan el constructor de base de conocimiento con medios fáciles para describir los tipos de objetos de dominio que el sistema debe de modelar. La descripción de un tipo de objeto puede contener la descripción del prototipo de objetos individuales de ese tipo; estos prototipos pueden ser usados para crear una descripción por defecto de un objeto cuando su tipo se vuelve desconocido en el modelo.

Un marco provee una representación estructurada de un objeto o clase de objetos. Por ejemplo, un marco puede representar un automóvil, y otro una clase completa de automóviles. Las construcciones están disponibles en un lenguaje de marcos para organizar los marcos que representan clases en taxonomías. Estas construcciones permiten a un diseñador de base de conocimiento(o ingeniero del K) describir cada clase como una especialización o subclase de otra más genérica. Así los automóviles pueden ser descritos como vehículos más un conjunto de propiedades que distinguen a los autos de otro conjunto de vehículos[20].

Cada marco se caracteriza por un conjunto de *campos* o *"slots"* que sirven para identificar los marcos, estos están especialmente concebidos para tareas de reconocimiento. El mecanismo es el siguiente, la información recibida hace que se activen unos marcos y esta a su vez provoca la activación de otros marcos conectados con los primeros, dando lugar así a una *red de activación*, cuyo objetivo es predecir y explicar la información que se va a encontrar en esa situación. Este reconocimiento basado en expectativas se denomina a veces *reconocimiento descendente*. Otra de las ideas novedosas de Minksy es la posibilidad de tener distintos marcos para definir una misma entidad desde distintos *puntos de vista*. El propósito de Minsky consistía en dar sugerencias e ideas más que en concretar los detalles de un cierto tipo de representación[45].

Las ventajas de los lenguajes de marcos son considerables: capturan la manera en la que típicamente piensan los expertos sobre mucho de su conocimeinto, facilita una representación estructural y consisa de relaciones útiles, y soporta una técnica consisa de *definición-por-especialización* que es fácil de usar para la mayoría de los expertos. Además se han desarrollado algoritmos de deducción de propósito especial que aprovechan las características estructurales de los marcos para realizar rápidamente un conjunto de inferencias comúnmente necesitadas en sistemas basados en conocimeinto[20].

Los marcos son básicamente principios de Programación Orientada a Objetos (POO) pero aplicados en IA, porque su desarrollo se soporta con la idea básica que la conducta humana se caracteriza por tomar estándares y proceder en situaciones familiares para generar conocimeinto a partir de inferencias.

6.2. Objetos

Los objetos, son similares a los marcos. Ambos sirven para agrupar conocimiento asociado, soportan herencia, abstracción y el concepto de procedimientos agregados. La diferencia radica en lo siguiente:

1. En los marcos, a los programas y a los datos se los trata como dos entidades relacionadas separadas. En cambio en los objetos se crea una fuerte unidad entre los procedimientos (métodos) y los datos.

2. Los demonios de los marcos sirven sólo para computar valores para las diversas ranuras o para mantener la integridad de la base de conocimientos cada vez que una acción de algún marco, afecta a otro. En cambio, los métodos utilizados por los objetos son más universales ya que proporcionan cualquier tipo general de computación requerida y además soportan encapsulamiento y polimorfismo.

Un objeto es definido como una colección de información representando una entidad del mundo real y una descripción de cómo debe ser manipulada esta información, esto es los métodos. Es decir, un objeto tiene un nombre, una caracterización de clase, varios atributos distintivos y un conjunto de operaciones. La relación entre los objetos viene definida por los mensajes. Cuando un objeto recibe un mensaje válido, responde con una acción apropiada, retornando un resultado. Las propiedades fundamentales de los objetos son las siguientes[6]:

- **Clase**. Definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ella.

- **Objeto**. Instancia de una clase. Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos), los mismos que consecuentemente reaccionan a eventos. Se corresponden con los objetos reales del mundo que nos rodea, o con objetos internos del sistema.

- **Método**. Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un .^{ev}entoçon un nuevo mensaje para otro objeto del sistema.

- **Atributos**. Características que tiene la clase.

- **Mensaje**. Una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.

- **Propiedad**. Contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.

Además de estás propiedades básicas cuentan con propiedades muy útiles entre las cuales las más básicas se listan a continuación[6].

- **Abstracción**. Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar "cómo" se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos, y, cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción. El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real.

- **Encapsulamiento**. Significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión (diseño estructurado) de los componentes del sistema. Algunos autores confunden este concepto con el principio de ocultación, principalmente porque se suelen emplear conjuntamente.
- **Polimorfismo**. Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre; al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O, dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama asignación tardía o asignación dinámica.
- **Herencia**. Las clases no se encuentran aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento, permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Cuando un objeto hereda de más de una clase se dice que hay herencia múltiple.

Los objetos, como forma de representación del conocimiento ofrece las siguientes ventajas:

1. Poder de abstracción.
2. Encapsulamiento o capacidad de esconder información.
3. Herencia, es decir pueden recibir características o propiedades de sus ancestros.
4. Polimorfismo, que permite crear una interfaz común para todos los diversos objetos utilizados dentro del dominio.
5. Posibilidad de reutilización del código.
6. Mayor facilidad para poder trabajar eficientemente con sistemas grandes.

Las desventajas son similares a las que se indicaron para las plantillas:

1. Dificultades para manejar objetos que se alejan demasiado de la norma.
2. Dificultades para manejar situaciones que han sido encontradas previamente.

7. Representación del conocimiento basada en restricciones

La idea básica en la programación por restricción es que el usuario establezca limitaciones utilizando un solucionador de restricciones de propósito general para resolverlas. Las restricciones son solo relaciones. Formalmente, un problema de satisfacción de restricciones consiste en un conjunto de variables, cada una con un dominio de valores, y un conjunto de relaciones en los subconjuntos de estas variables.

7.1. Representación por restricciones

Si bien la definición de un conjunto de restricciones puede parecer una forma sencilla de modelar un problema, encontrar un buen modelo que funciona bien con un programa de solución elegida no es fácil. Se piensa en restricciones como inecuaciones sobre objetos arbitrarios, estas restricciones son usadas para codificar las condiciones del sistema.

Algunos dominios de aplicación de este modelo son:

- Dominios booleanos, donde solo existen restricciones del tipo verdadero/falso.
- Dominios en variables enteras y racionales.
- Dominios lineales, donde sólo se describen y analizan funciones lineales.
- Dominios finitos, donde las restricciones son definidas en conjuntos finitos.
- Dominios mixtos, los cuales involucran dos o más de los anteriores.

Entre las características de este modelo[13] están la transparencia referencial, las unidades pequeñas de código tienen sus propios significados aislados de los demás, más que un pareo de patrones se implementa paso de parámetros de manera bidireccional, no es necesario asignar memoria de manera explícita, las estructuras de datos son fáciles de construir.

Algunas de las inconveniencias de este modelo son[13]: falta de modularidad, es difícil crear restricciones de manera dinámica ya que tienen que estar estáticamente definidas, se puede dar el caso en el que las restricciones del sistema no sean suficientemente expresivas para reflejar el problema completo, o se pueden dar soluciones en un formato no deseado.

Una restricción global es una restricción sobre una secuencia de variables. Las restricciones globales suelen venir con un algoritmo de propagación de restricciones que realiza una mejor poda o más barata que si tratamos de expresar la restricción global en forma de relaciones más pequeñas. El ejemplo canónico de una restricción global es la restricción all-different . Una restricción “all-different” es un conjunto de estados de las variables en las que los valores de las variables debe ser diferente en todas. La restricción “all-different” es ampliamente utilizada en la práctica y debido a su importancia se ofrece como un “built-in en” la mayoría de paquetes de programación por restricciones comerciales.

En una restricción, un valor para una variable en la restricción se dice que tiene un soporte si existen valores para otras variables en la restricción de tal manera que la restricción se cumple. Una restricción es arco consistente si todos los valores en los dominios de las variables de la restricción tienen un apoyo. Una restricción se puede hacer de arco consistente en varias ocasiones eliminando los valores no admitidos de los dominios de las variables. La extracción de valores no admitidos se refiere a menudo como la poda de los dominios. Por ejemplo, sea 0, 1, 2 el dominio de las variables x y y y considerar la restricción $x + y = 1$. Hacer cumplir la consistencia de arco en esta restricción podaría el dominio de ambas variables a sólo 0, 1. Los valores podados de los dominios de las variables son inconsistentes a nivel local por lo que no puede ser parte de alguna solución. Hacer

cumplir la consistencia de arco nos obliga a recorrer cíclicamente el dominio hasta llegar a un punto fijo en donde todos pueden cumplir.

El poder de las restricciones globales es doble. En primer lugar, las restricciones globales facilitar la tarea de modelar como un problema basado en restricciones. En segundo lugar, para fines específicos se pueden diseñar algoritmos de propagación de restricciones que aprovechen la semántica de la restricción y por lo tanto serán mucho más eficientes. Recordando que la aplicación de la consistencia de arco en una restricción arbitraria es de $O(rd^r)$ en el peor de los casos, donde r es la aridad de la restricción y d es el tamaño de los dominios de las variables. Ahora, la restricción "all-different" puede calcular la consistencia de arco en orden $O\left(r^2d\right)$ en el peor de los casos y se pueden hacer compatibles los límites en $O(r)$.

7.2. Programación por restricciones

Los solucionadores de restricciones buscan el espacio de solución de forma sistemática ya sea por rastreo, ramificaciones y algoritmos limitantes. Los métodos sistemáticos de búsqueda a menudo son la interpolación y la inferencia, la inferencia que consiste en la propagación de la información contenida en una restricción a las restricciones vecinas. Tal inferencia reduce el espacio de búsqueda. La propagación de procedimientos especiales puede ser diseñada para adaptarse a restricciones específicas (llamadas restricciones globales), que se producen a menudo en la vida real. Tales restricciones globales son un componente importante en el éxito de la programación por restricciones. Proporcionan pautas comunes para ayudar a los usuarios a modelar los problemas. También facilitan la búsqueda de una solución más eficiente y más eficaz.

Aunque los problemas basados en restricciones son en general NP-completo, hay clases importantes que se pueden resolver en tiempo polinomial. Se identifican por el uso común de variables entre restricciones, o por el lenguaje para definir las restricciones. Por ejemplo, los problemas basados en restricciones con un gráfico de conectividad en forma de árbol se pueden resolver en tiempo polinomial.

Un ejemplo de un lenguaje de restricciones aplicado a un contexto concurrente sería el siguiente[34], imagina un sistema que consiste de procesos concurrentes interactuando por medio de datos compartidos en alguna base de datos. Los datos compartidos pueden pensarse como una colección de afirmaciones en algún fragmento en una lógica de primer orden $(L; \vdash)$. Los procesos se comunican agregando información al "pool" común de datos (una operación de "aviso"), o preguntando si una afirmación es implicada por el "pool" existente de datos (una operación de "pregunta"). Así un lenguaje de restricciones concurrente incluiría:

- Un lenguaje de almacenamiento de datos $(L; \vdash)$ para describir las afirmaciones que se pueden hacer(el sistema de restricciones.
- Un proceso de pregunta-aviso para describir cómo los procesos interactuan con el "pool" de datos.

Por analogía con un lenguaje imperativo podemos llamar a al colección de afirmaciones compartidas(o restricciones) el "almacén". En contraste con la programación imperativa, sin embargo un "almacén" podría dar información parcial sobre los valores de las variables. Por ejemplo más que decir que una variable tiene el valor 1994, podría apenas decir que la variable está entre 1729 y 4204. El lenguaje de pregunta-aviso será un simple proceso de cálculo, basado en las nociones primitivas de pregunta y respuesta. la operación de aviso es el mecanismo para la comunicación: toma una restricción ϕ y la agrega al "pool" común de datos. La construcción de pregunta es el mecanismo para la sincronización, dada una restricción ϕ, pregunta(ϕ) tiene éxito o falla dependiendo de si el

almacén implica ϕ. En el caso exitoso, el proceso continua, de otra forma el proceso se suspende hasta que (si alguna vez sucede) mas datos están disponibles. La construcción pregunta es lo que le da al programador la habilidad de manipular información parcial ya que uno puede consultar el almacén a pesar de que la información está parcialmente definida.

Un solucionador de restricciones pueden implementarse en cualquier lenguaje. Sin embargo, hay lenguajes, especialmente diseñado para representar las relaciones de restricción. Estos lenguajes están basados en lógica, son imperativos, orientados a objetos, o basados en reglas.

Los lenguajes de programación por restricciones son típicamente ampliaciones de otro lenguaje. El primer lenguaje utilizado a tal efecto fue Prolog. Por esta razón es que este campo fue llamado inicialmente Programación Lógica con Restricciones.

Desde el punto de vista práctico, la importancia de este tipo de lenguajes se debe en buena medida a la posibilidad de abordar declarativamente la resolución de problemas combinatorios discretos (generalmente NP-completos) que aparecen en muchas áreas de aplicación tales como planificación, diagnóstico de fallas, análisis y diseño de circuitos, verificación del hardware, control de tráfico o biología molecular, por citar sólo algunas de las más comunes que usan estas herramientas. Algunos lenguajes de programación con restricciones son:

- **B-Prolog** (Basado en Prolog, proprietario)
- **CHIP V5** (Basado en Prolog, también existen bibliotecas en C y C++, proprietario)
- **Ciao Prolog** (Basado en Prolog, software libre: GPL/LGPL)
- **ECLiPSe** (Basado en Prolog, proprietario)
- **Mozart** (Basado en Oz, software libre: X11)
- **SICStus** (Basado en Prolog, propietario)
- **GNU Prolog** (Basado en Prolog, software libre)
- **SWI Prolog** Un entorno Prolog que contiene varias librerías para soluciones con restricciones (LGPL)

Uno de los conceptos más importantes en la programación por restricciones es la consistencia local. Una inconsistencia local es la instancia de algunas de las variables que satisface una restricción, pero no puede extenderse a una o más restricciones adicionales y por lo tanto no puede ser parte de cualquier solución. Si estamos utilizando una búsqueda por rastreo para encontrar una solución, tales inconsistencias son callejones sin salida en la búsqueda y causando esfuerzos inútiles en la búsqueda.

La técnica principal para resolver los problemas basados en restricciones es la búsqueda. Un algoritmo de búsqueda para resolver un problema basado en restricciones puede ser completo o incompleto. Los algoritmos de búsqueda completa o sistemática, garantizan que si existe una solución se encontrará, y pueden ser utilizados para demostrar que problema no tiene una solución y encontrar una solución óptima. Búsqueda incompleta o no sistemática, no se puede utilizar para mostrar que un problema tiene una solución, sin embargo, este tipo de algoritmos suelen ser eficaces en la búsqueda de una solución si es que existe y se pueden utilizar para buscar una aproximación a una solución óptima. Los algoritmos de "backtrack" son, en general, los ejemplos de algoritmos de búsqueda completa y los algoritmos de búsqueda local son ejemplos de algoritmos de búsqueda incompletos.

En resumen la programación por restricciones es una herramienta que permite solucionar problemas de asignación de recursos de manera óptima estableciendo límites en con una metodología general para resolver estas restricciones.

8. Modelos evolutivos y no deterministas

La computación evolutiva (CE) es una de las ramas de la Inteligencia Artificial que se aplica para la resolución de problemas de optimización. La CE está inspirada en los mecanismos de evolución biológica propuestos por Darwin, Medel y Lamark. Se puede decir que la inteligencia artificial es una rama que trata de "imitar" a la inteligencia natural y por tanto la computación evolutiva se basa en imitar la evolución biológica tal y como la entendemos hoy en día. La CE trata de resolver problemas de optimización combinatoria, con el fin de encontrar el mejor resultado al problema, es decir tal y como pasa en la naturaleza, ya que el gran problema que hay, sobre todo en el entorno animal; es la supervivencia y solo los más fuertes y mejor adaptados al entorno son capaces de sobrevivir y reproducirse; en otras palabras, son los mejores individuos posibles o la mejor solución al problema. El CE toma algunos conceptos de la naturaleza para inspirar su metodología estos conceptos son:

- Una población de individuos coexiste en un determinado entorno con recursos limitados.
- La competición por los recursos provoca la selección de aquellos individuos que están mejor adaptados al entorno.
- Estos individuos se convierten en los progenitores de nuevos individuos a través de procesos de mutación y recombinación.
- Los nuevos individuos pasan a competir por su supervivencia.
- Con el paso del tiempo, esta selección natural provoca el incremento en la "calidad" de los individuos de la población.

8.1. Redes neuronales artificiales

Las redes neuronales (también conocidas como sistemas conexionistas) son un modelo computacional basado en un gran conjunto de unidades neuronales simples (neuronas artificiales), de forma aproximadamente análoga al comportamiento observado en los axones de las neuronas en los cerebros biológicos. Cada unidad neuronal está conectada con muchas otras y los enlaces entre ellas pueden incrementar o inhibir el estado de activación de las neuronas adyacentes. Cada unidad neuronal, de forma individual, opera empleando funciones de suma. Puede existir una función limitadora o umbral en cada conexión y en la propia unidad, de tal modo que la señal debe sobrepasar un límite antes de propagarse a otra neurona. Estos sistemas aprenden y se forman a sí mismos, en lugar de ser programados de forma explícita, y sobresalen en áreas donde la detección de soluciones o características es difícil de expresar con la programación convencional[7], una red neural artificial es un grupo interconectado de nodos similar a la vasta red de neuronas en un cerebro biológico. Cada nodo circular representa una neurona artificial y cada flecha representa una conexión desde la salida de una neurona a la entrada de otra(Figura 3).

8.2. Algoritmos genéticos

Una de las áreas más prometedoras bajo rápido desarrollo son los algoritmos genéticos(AG). La ventaja más significativa de usar AGs recae en la flexibilidad y la adaptabilidad de la tarea, en combinación con desempeño robusto y características de desempeño de búsqueda global. De hecho los AGs deben de ser entendidos como un concepto general adaptable para reslver problemas, especialmente para resolver problemas difíciles de optimización. Los algoritmos genéticos a pesar de estar inspirados en un proceso evolutivo tienen características que no están ligadas al contexto del cómputo evolutivo, estas características son[30]:

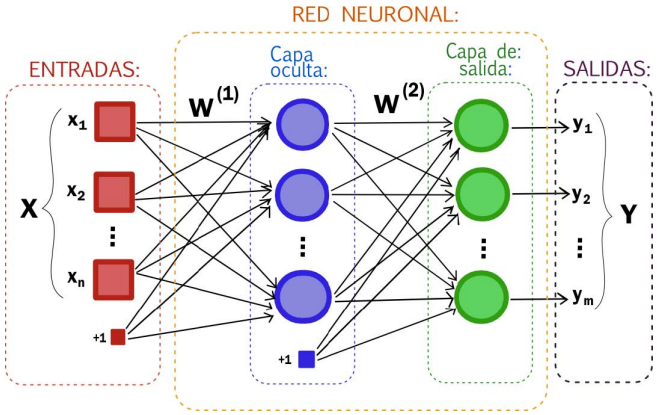

Figura 3: Arquitectura general de una RNA [8]

1. Trabaja en un espacio n-dimensional discreto D definido en N^D más que en R^D.
2. Explora D buscando una aproximación del vector óptimo de (1) analizando simultaneamente un conjunto finito $S(t)$ de soluciones candidatas.
3. Los elementos de $S(t) = \{s_1(t), s_2(t), ..., s_n(t)\}$ están explícitamente codificados de una manear adecuada.
4. La información referente a la adecuación parcial de los elementos en $S(t)$ es extraida resolviendo el problema para todos los $s_i(t)$.
5. Los elementos calificados de $S(t)$ son analizados para seleccionar un subconjunto de ellos que es considerado más apropiado para mejorar la búsqueda en el espacio del problema.
6. Secciones de los códigos de $s_i(t)$ son selectivamente combinados para acelerar la convergencia del algoritmo.
7. Elementos seleccionados del código son alterados periódica y aleatoriamente.
8. Un subconjunto de las mejores soluciones de $S(t)$ es retenida durante los pasos del AG.
9. El algoritmo se cicla en los pasos 4-8 hasta que se encuentra con una condición de paro.

8.3. Mapas cognitivos

Los mapas cognitivos (también conocidos como mapas mentales, modelos mentales o modelos cognitivos) son un tipo de procesamiento mental compuesto por una serie de transformaciones psicológicas mediante las cuales un individuo puede adquirir, cifrar, almacenar, recuperar y descifrar información sobre las localizaciones relativas y los atributos de los fenómenos en su entorno espacial cotidiana o metafórico. En términos más simples, los mapas cognitivos son un método que usamos para construir y acumular conocimientos espaciales, lo que permite que el "ojo de la mente" visualice imágenes con el fin de reducir la carga cognitiva, mejorar la recuperación y el aprendizaje de la información. Este tipo de pensamiento espacial también puede ser utilizado como una metáfora para las tareas no-espaciales donde las personas que realizan tareas no espaciales que implican memoria e imágenes utilizan el conocimiento espacial para ayudar en el procesamiento de la tarea. El más antiguo método formal conocido para recordar datos utilizando una localización espacial es el "método de loci". Para usarlo primero hay que memorizar el aspecto de un lugar físico (por ejemplo, la secuencia de habitaciones en un edificio). Cuando una lista de palabras, por ejemplo, debe ser memorizada, el aprendiz visualiza un objeto que representa esa palabra en uno de los lugares prememorizados. Para recuperar la lista, el alumno mentalmente "camina a través de" los lugares memorizados, observando los objetos colocados allí durante la fase de memorización.

8.4. Estrategias Bioinspiradas

Básicamente, lo que se hace es tomar como modelo un fenómeno que se encuentra en el medio ambiente, por ejemplo, comportamientos sociales de seres vivos simples que en conjunto resuelven problemáticas muy complejas, tales como las colonias de hormigas y abejas, emular estas conductas en la computadora para crear un programa de optimización, que encuentre el mejor resultado ante una determinada dificultad en la vida del ser humano, tal como lo harían estos organismos. Esta área de estudio surgió a través del cómputo evolutivo, que toma como margen la teoría darwinista para crear técnicas que emulen la evolución de las especies y la supervivencia del más apto[9].

8.4.1. Colonia de Hormigas

Este algoritmo es un miembro de la familia de los métodos de inteligencia de enjambres. Inicialmente propuesto por Marco Dorigo en 1992 en su tesis de doctorado[31], el primer algoritmo surgió como método para buscar el camino óptimo en un grafo, basado en el comportamiento de las hormigas cuando estas están buscando un camino entre la colonia y una fuente de alimentos. La idea original se ha diversificado para resolver una amplia clase de problemas numéricos, y como resultado, han surgido gran cantidad de problemas nuevos, basándose en diversos aspectos del comportamiento de las hormigas.

8.4.2. Enjambre de abejas

El algoritmo de colonia de abejas (ABC) es un algoritmo de metaheurística introducida por Karaboga en 2005[27] ,y simula el comportamiento de alimentación de las abejas melíferas. El algoritmo ACA tiene tres fases: la abeja empleada, la abeja espectador y la exploradora. En la abeja empleada y las fases de abejas espectador,las abejas explotan las fuentes de búsquedas locales en la vecindad de las soluciones seleccionadas sobre la base de la selección determinista en la fase de abeja ocupada y la selección probabilística en la fase de abeja espectador. En la fase de abeja exploradora que es una analogía de abandono de las fuentes de alimentos agotados en el proceso de búsqueda de alimento, las soluciones que no son beneficiosos para el progreso de la búsqueda ya están abandonados, y nuevas soluciones se insertan en lugar de ellos para explorar nuevas regiones del espacio de búsqueda. El algoritmo tiene una capacidad equilibrada de exploración-explotación.

8.4.3. Búsqueda armónica

En el modelo básico de BA, cada solución candidata es considerada una “armonía”, y es representada por un vector n-dimensional[14]. La población inicial de las soluciones candidatas es aleatoriamente inicializada y almacenada dentro de una memoria (Harmony Memory ”HM“). De esta manera, una nueva solución es generada a partir de uno de los elementos contenidos en HM, a través de una operación de re-inicialización aleatoria o mediante una operación de ajuste de ”tono“ de un vector contenido en ella. Finalmente, la HM es actualizada mediante la comparación de la nueva solución candidata y el peor de los vectores contenidos en HM, si esta es mejor, remplazará el lugar del vector dentro de la memoria, de lo contrario no existirá cambio alguno. Este proceso se realiza hasta cumplir el criterio de paro. La forma básica del algoritmo BA consiste en tres etapas: inicialización, improvisación de nuevas armonías (generación de nuevas soluciones) y la actualización de la memoria (HM).

8.5. Modelos cognitivos de la memoria

8.5.1. Modelo de Atkinson – Shiffrin

Este modelo se basa en el trabajo de otros para distinguir entre el almacenamiento y uso de la memoria a corto plazo y el almacenamiento y uso de la memoria a largo plazo. Sin embargo, Atkinson y Shiffrin agregaron otro componente en lo que denominaron memoria sensorial y reconceptualizaron a los recuerdos como memorias de corto plazo y largo plazo. En esencia, este modelo afirma que el input ambiental se procesa a través de los sentidos y posteriormente se traslada a la memoria de corto plazo o a la memoria de trabajo temporal.

8.5.2. Modelo de Tulving

En este modelo se marca una diferencia entre la memoria episódica y otros tipos de sistemas de aprendizaje y de memoria en el cerebro, Tulving las divide en 3 categorías: memoria procedural (*know-how*), memoria semántica(o genérica) que almacena los conocimientos generales que no implican ninguna estructura de eventos, y la memoria episódica, que es el recuerdo de eventos temporales en el tiempo subjetivo ("lo hice", "voy a hacer esto").

8.5.3. Proceso Paralelo Distribuido (Rumelhard y McClelland)

Son una clase de modelos de procesamiento de información inspirados en las neuronas que tratan de modelar el procesamiento de información en la forma en que realmente se lleva a cabo en el cerebro.Tiene tres principios básicos: (a) representación distribuida de información, (b) la memoria y el conocimiento se almacenan de manera implícita en las conexiones, (c) el aprendizaje ocurre con cambios en las fuerzas de las conexiones.

8.6. Imagen y percepción mental (reconocimiento de patrones y liguística)

8.6.1. Percepción visual 3D (Teoría de David Marr)

Marr propuso la idea de que hay que entender los sistemas de procesamiento de información en tres niveles diferentes y complementarios de análisis. Esta idea se conoce en la ciencia cognitiva como la hipótesis tri-nivel de Marr[28].

8.6.2. Percepción lingüística

La percepción del lenguaje se ocupa de los procesos de extracción de información de la señal acústica o, alternativamente, gráfica. Por tanto, se ocupa tanto de los procesos iniciales de análisis de la señal, como de los procesos más complejos donde el análisis es sintáctico/semántico. No obstante, percepción del lenguaje se refiere, habitualmente, a los procesos iniciales incluido el reconocimiento de la palabra y se reserva el término comprensión del lenguaje para los procesos más complejos.

9. Modelos para el manejo de Incertidumbre e Imprecisión

Existen varias causas de incertidumbre que tienen que ver con la información, el conocimiento y la representación. La información puede ser incompleta, poco confiable o presentar ruido o distorsión, esto puede ocasionar que el conocimiento se vuelva de naturaleza imprecisa o contradictoria, además la representación se vuelve poco adecuada y falta de poder descriptivo. Los efectos de esto es que se pierden varias propiedades de los sistemas que no tienen incertidumbre, basados en lógicas o reglas, lo cual hace el manejo de incertidumbre más complejo. Las principales dos características que, en general, ya no aplican son la modularidad y la monotonicidad[36].

9.1. Modelos estadísticos - probabilistas

Actualmente la teoría de la probabilidad se ha desarrollado y extendido enormemente gracias a muchos pensadores que han contribuido a su crecimiento, y es, sin duda, una parte importante y bien establecida de las matemáticas. La teoría de la probabilidad ha resultado muy útil para modelar fenómenos de muy diversas disciplinas del conocimiento humano en donde es necesario incorporar la incertidumbre o el azar como un elemento esencial del modelo. Así la probabilidad puede definirse como aquella parte de las matemáticas que se encarga del estudio de los fenómenos aleatorios[39]. La probabilidad de un evento A es un número real en el intervalo $[0, 1]$ que se denota por $P(A)$ y representa una medida de la frecuencia con la que se observa la ocurrencia de este evento cuando se efectúa el experimento aleatorio en cuestión. Existen definiciones específicas de la probabilidad dependiendo del enfoque se pueden encontrar 3 escuelas principales: clásica, frecuentista y bayesiana. La primera dice que la probabilidad de cualquier evento en el espacio de eventos es igual, la frecuentista dice que para un número infinito de experimentaciones, se obtiene el valor real de la probabilidad de un evento, por último la más aceptada es la bayesiana, que dice que las probabilidades se van actualizando con evidencia nueva. Este último está basado en el teorema de bayes (1)

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \tag{1}$$

De este teorema (1) surgen dos conceptos importantes, probabilidad conjunta $P(A, B)$, que es la probabilidad de que dos eventos distintos tomen ciertos valores; y la probabilidad condicional que cuantifica la probabilidad de un evento una vez que sabemos que otro ocurrió $P(B \mid A)$ y se lee probabilidad de B dado el evento A. A partir del teorema de bayes se derivan dos reglas importantes para el cálculo de probabilidades, la regla del producto (3) y la regla de la suma (2)

$$P(A = a_i) = \sum_{j=1}^{L} P(A = a_i, B = b_j) \tag{2}$$

$$P(A = a_i, B = b_j) = \frac{n_{ij}}{N} = P(B = b_j \mid A = a_i)P(A = a_i) \tag{3}$$

9.2. Modelos aproximados(Factor de certeza)

Son métodos en general heurísticos, que permiten obtener descripciones e inferencias estructurales de calidad suficiente para muchas aplicaciones y que en general sí se pueden realizar en cualquier computador. Debido a la limitación que se tiene en la mayoría de los casos prácticos de no disponer de una gran cantidad de datos históricos, obligó al desarrollo de un método de inferencia aproximado, que en general proporcione resultados exactos aún cuando estén basados en datos limitados. A

este se lo denominó el formalismo del **factor de certeza**. Los resultados son más o menos ciertos en función de la certeza o falsedad de los hechos y conocimientos utilizados. Los factores de certeza que van asociados a los predicados, por lo general pueden ir de 0 a 100. El procedimiento a seguirse en el modelo aproximado es el siguiente:

- El factor de un conjunto de condiciones unidas por el operador lógico Y (AND) es igual al mínimo de los factores que intervienen.
- El factor de un conjunto de condiciones unidas por el operador lógico O (OR) es igual al máximo de los factores que intervienen.
- El factor de certeza de una conclusión es igual al producto del factor de certeza de las condiciones por el de la regla.

El formalismo del factor de certeza es uno de los más populares en los sistemas basados en conocimientos, debido a los siguientes puntos:

1. Es un modelo computacional simple que permite a los expertos estimar la confianza en las conclusiones derivadas.
2. Permite expresar el grado de creencia en cada hipótesis, permitiendo la expresión del efecto de múltiples fuentes de evidencia.
3. Permite la captura de conocimiento en una regla, incluyendo la cuantificación de la incertidumbre.
4. La asignación de valores de certeza es simple aunque subjetiva. No se requiere base estadística, simplemente se solicita al experto su estimación.

También han sido puntualizadas ciertas dificultadas asociadas a los factores de certeza:

1. Los valores de certeza no pueden representar en forma eficiente y natural ciertas dependencias entre estimaciones de incertidumbre.
2. El valor del factor de certeza asociado a una regla es dependiente de la fuerza de la asociación entre las premisas y las conclusiones. A medida que mayor conocimiento sea descubierto y añadido o retirado de una base de conocimiento, obligará a que los valores de certeza del conocimiento existente tengan que variar. Esto hará que los cambios involucrados en la base de conocimiento sean bastante complejos.

9.3. Modelos de lógica difusa (borrosa)

La lógica difusa (también llamada lógica borrosa) se basa en lo relativo de lo observado como posición diferencial. Este tipo de lógica toma dos valores aleatorios, pero contextualizados y referirlos entre sí. Así, por ejemplo, una persona que mida dos metros es claramente una persona alta, si previamente se ha tomado el valor de persona baja y se ha establecido en un metro. Ambos valores están contextualizados a personas y referidos a una medida métrica lineal[10]. Las reglas involucradas en un sistema borroso, pueden ser aprendidas con sistemas adaptables que aprenden al "observar" como operan las personas los dispositivos reales, o estas reglas pueden también ser formuladas por un experto humano. En general la lógica borrosa se aplica tanto a sistemas de control como para modelar cualquier sistema continuo de ingeniería, física, biología o economía. La lógica borrosa es entonces definida como un sistema matemático que modela funciones no lineales, que convierte unas

entradas en salidas acordes con los planteamientos lógicos que usan el razonamiento aproximado. Por medio de la Lógica Difusa pueden formularse matemáticamente nociones como "un poco caliente" o "muy frío", para que sean procesadas por computadoras y cuantificar expresiones humanas vagas, tales como "muy alto" o "luz brillante". De esa forma, es un intento de aplicar la forma de pensar humana a la programación de las computadoras.

En la teoría estándar de conjuntos, un objeto es miembro del conjunto o no lo es. No existen posibilidades intermedias. En la lógica difusa, se generaliza este concepto, permitiendo que las funciones características de confianza asuman valores reales, dentro del intervalo $[0, 1]$, (Totalmente FALSO = 0; totalmente VERDADERO = 1). Estos valores indican el grado o nivel de pertenencia del objeto dentro del conjunto difuso. La teoría de los conjuntos difusos permite que un elemento sea parcialmente miembro de un determinado conjunto.

Operaciones con conjuntos difusos.

Sean A y B dos conjuntos difusos del universo U, y $p_A(x), p_B(x)$ los grados de pertenencia de x en A y B.

Unión	$B = \{MAX \cup A(p_A(x), p_B(x)) \mid x \in U\}$
Intersección	$B = \{MIN \cap A(p_A(x), p_B(x)) \mid x \in U\}$
Complemento	$A^C = \{(1 - p_A(x)\} \mid x \in U$
Normalización	$NORM(A) = \{p_A(x)/MAXp_A(y) \mid x, y \in U\}$
Dilatación	$DIL(A) = \{\sqrt{p_A(x)} \mid x \in U\}$
Concentración	$CON(A) = \{p_A(x)^2 \mid x \in U\}$

Tabla 5: Tabla de operadores para conjuntos difusos

10. Ontologías

Un cuerpo de conocimiento formalmente representado se basa en una conceptualización: los objetos, conceptos, y otras entidades que se supone que existen en algún área de interés ("nube") y la relaciones que mantienen entre ellos [21]. Una conceptualización es una visión abstracta y simplificada del mundo que queremos representar con algún propósito. Cada base de conocimiento, sistema basado en el conocimiento, o agente de conocimiento está relacionado con una cierta conceptualización, explícita o implícitamente.

10.1. Definición

En ciencias de la computación y ciencias de la comunicación, una ontología es una definición formal de tipos, propiedades, y relaciones entre entidades que realmente o fundamentalmente existen para un dominio de discusión en particular. Es una aplicación práctica de la ontología filosófica, con una taxonomía[5].

Una ontología cataloga las variables requeridas para algún conjunto de computación y establece las relaciones entre ellos[22]. En los campos de la inteligencia artificial, la Web Semántica, ingeniería de sistemas, ingeniería de software, informática biomédica, bibliotecología y arquitectura de la información se crean ontologías para limitar la complejidad y para organizar la información. La ontología puede entonces ser aplicada para resolver problemas.

Los componentes más comunes de una ontología son[5]:

- Individuos: instancias u objetos (lo básico u objetos de "bajo nivel")
- Clases: conjuntos, colecciones, conceptos, clases en programación, tipos de objetos,o tipos de cosas.
- Atributos: aspectos, propiedades, rasgos, características, o parámetros que objetos (y clases) pueden tener.
- Relaciones: formas en la cual clases y los individuos se pueden relacionar unos con otros .
- Funciones: Complejas estructuras formadas de cierta relación que pueden ser usada en lugar de un término individual en una declaración
- Restricciones: establecen descripciones formales de lo que debe ser verdad con el objetivo de que alguna aserción pueda ser aceptada como entrada.
- Reglas: Declaraciones con forma de oraciones si-entonces (antecedente-consecuente) que describen inferencias lógica que puede ser derivables de una aserción en una forma particular .
- Axiomas: aserciones (incluyendo reglas) en una forma lógica que juntos incluyen toda la teoría que la ontología describe en su dominio de aplicación. Esta definición es diferente de los "axiomas" en gramáticas generadas y forma lógica. En esas disciplinas, axiomas solamente incluyen declaraciones especificadas como un conocimiento a priori. En las ontologías, ."xiomas"también incluyen teorías derivadas de declaraciones axiomáticas.
- Eventos: los cambios de los atributos o relaciones.

Cuando decidimos cómo representar algo en una ontología, estamos tomando decisiones de diseño. Para orientar y evaluar nuestros diseños, necesitamos criterios objetivos que se basen en el propósito del artefacto resultante, en lugar de basarlos sobre nociones a priori de naturalidad o de Verdad. Aquí se propone un conjunto preliminar de criterios de diseño de ontologías, cuya finalidad es el intercambio de conocimientos y la interoperabilidad entre los programas sobre la base de una conceptualización compartida.

- Claridad: Una ontología debe comunicar efectivamente el significado de los términos definidos. Las definiciones deben ser objetivas. Si bien la motivación para la definición de un concepto podría surgir de situaciones sociales o de requisitos de cómputo, la definición debe ser independiente del contexto social o de cómputo. El formalismo es un medio para este fin. Cuando una definición pueda ser expresada en axiomas lógicos, entonces debería serlo. Siempre que sea posible, una definición completa (un predicado definido por condiciones necesarias y suficientes) es preferible a una definición parcial (que se define sólo por condiciones necesarias o suficientes). Todas las definiciones deben ser documentados con lenguaje natural.

- Coherencia: Una ontología debe ser coherente, es decir, se deben sancionar las inferencias que sean coherentes con las definiciones. Por lo menos, los axiomas de la definición deben ser lógicamente consistentes. La coherencia debe aplicarse también a los conceptos que se definen de manera informal como aquellos descritos en la documentación y ejemplos mediante el uso de lenguaje natural. Si una frase que se puede deducir de los axiomas contradice una definición o ejemplo dado de manera informal, entonces, la ontología es incoherente.

- Extensibilidad: Una ontología debe estar destinada a anticipar los usos del vocabulario compartido. Debería ofrecer una base conceptual para una serie de tareas previstas, y la representación debe ser diseñada de manera que se pueda ampliar y especializar la ontología en forma monótona. En otras palabras, uno debe ser capaz de definir nuevos términos para usos especiales basados en el vocabulario existente de una manera que no requiera la revisión de las definiciones existentes.

- Mínimo sesgo de codificación: La conceptualización debe ser especificada en el nivel de conocimientos sin depender de un nivel de codificación simbólica especial. Un sesgo de odificación resulta cuando se toman decisiones basadas puramente en la conveniencia de notación o de implementación. El sesgo de codificación debe reducirse al mínimo, dado que los agentes que comparten conocimiento pueden ser implementados sobre diferentes sistemas y estilos de representación.

- Compromiso ontológico mínimo: Una ontología deberá exigir el mínimo compromiso ontológico suficiente para soportar las actividades de intercambio de conocimientos. Una ontología deberá hacer el menor número posible de afirmaciones sobre el mundo que está siendo modelado, permitiendo así a las partes comprometidas con la ontología la libertad de especializar y crear instancias de la ontología según sea necesario. Dado que el compromiso ontológico está basado en el uso consistente del vocabulario, el compromiso ontológico puede ser minimizado especificando la teoría más débil (permitiendo la mayoría de los modelos) y definiendo sólo aquellos términos que son esenciales para la comunicación del conocimiento consistente con esa teoría.

10.2. Aplicaciones y usos para el desarrollo de SBKs

Es un pensamiento compartido por la comunidad informática que las ontologías pueden ayudar a construir mejores sistemas software y con mayor interoperabilidad. Las características más

importantes que las ontologías proporcionan a la Ingeniería del Software son: (1) Una semántica y una taxonomía explícita; (2) Un enlace explícito entre conceptos y relaciones y teorías genéricas; (3) Ausencia de polisemia dentro de un contexto formal; (4) Modularización de contextos; (5) Axiomatización mínima para detallar diferencias entre conceptos similares; (6) Una buena política de elección de nombres; y (7) Una documentación rica. En Ingeniería del Software puede hacerse uso de ontologías a distintos niveles de generalidad. Por ejemplo, las ontologías a nivel de dominio son especialmente útiles para el desarrollo de software reutilizable de alta calidad, gracias a que las ontologías proveen una terminología no ambigua que puede ser compartida por todos los procesos de desarrollo.

Un ejemplo de aplicación de ontologías es MANTIS [41] es un "SE extendido", cuyo objetivo es la gestión del mantenimiento de software. Incluye una ontología, común a todos sus elementos, que tiene dos usos principales: (1) ayudar a la compartición del conocimiento sobre el mantenimiento entre todos los actores que intervienen en el proceso de mantenimiento de software; (2) filtrar el conocimiento, mediante el uso de modelos y meta-modelos, que por definición muestran sólo una parte de la realidad y que de esta manera ayudan a decidir, en el momento de construir los modelos (del nivel M1 de la arquitectura conceptual basada en MOF), que es lo que debe ser extraído de los sistemas reales. La ontología general está compuesta por 3 ontologías:

- Ontología de los Flujos de Trabajo (para los aspectos dinámicos)
- Ontología de la Medida (gestionar es medir)
- Ontología del Mantenimiento, que a su vez, se subdivide en 4 subontologías: de los Productos, de las Actividades, de Organización del Proceso, y de los Agentes.

Referencias

[1] Ejemplos de sistemas de expertos. http://sistemasexpertos2006.galeon.com/enlaces1463710.html. Accessed: 2017-05-31.

[2] Logica de temporal. https://es.wikipedia.org/wiki/L%C3%B3gica_temporal. Accessed: 2017-05-30.

[3] Logica de temporal anexo. http://disi.unal.edu.co/~lctorress/iartificial/IA0026l.pdf. Accessed: 2017-05-30.

[4] Metodología de desarrollo de software. https://es.wikipedia.org/wiki/Metodolog%C3%ADa_de_desarrollo_de_software. Accessed: 2017-05-30.

[5] Ontología. https://es.wikipedia.org/wiki/Ontolog%C3%ADa_(inform%C3%A1tica). Accessed: 2017-05-30.

[6] Programación orientada a obetos. https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos#Conceptos_fundamentales. Accessed: 2017-06-1.

[7] Redes neuronales artificiales. https://es.wikipedia.org/wiki/Red_neuronal_artificial. Accessed: 2017-05-30.

[8] Redes neuronales artificiales. http://elmundodelatecnologia18.blogspot.mx/2015/09/redes-neuronales-artificiales.html. Accessed: 2017-05-30.

[9] Redes neuronales artificiales. https://es.wikipedia.org/wiki/Sistemas_bioinspirados. Accessed: 2017-05-30.

[10] Redes neuronales artificiales. https://es.wikipedia.org/wiki/L%C3%B3gica_difusa. Accessed: 2017-05-30.

[11] Sistemas de producción. https://es.wikipedia.org/wiki/Sistemas_de_producci%C3%B3n. Accessed: 2017-05-30.

[12] Simon Blackburn. *The Oxford dictionary of philosophy*. OUP Oxford, 2005.

[13] Manuel Carro. Constraint logic programming. 1987.

[14] Erik Cuevas and Noé Ortega-Sánchez. El algoritmo de búsqueda armónica y sus usos en el procesamiento digital de imágenes. *Computación y Sistemas*, 17(4):543–560, 2013.

[15] Luis de Ledesma. Lógica para la computación. *Editorial RAMA*.

[16] Arjen P De Vries and Thomas Roelleke. Relevance information: a loss of entropy but a gain for idf? In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 282–289. ACM, 2005.

[17] Herbert Enderton and Herbert B Enderton. *A mathematical introduction to logic*. Academic press, 2001.

[18] Edward A Feigenbaum. The art of artificial intelligence. 1. themes and case studies of knowledge engineering. Technical report, DTIC Document, 1977.

[19] Alan Fern and Robert Givan. Online ensemble learning: An empirical study. *Machine Learning*, 53(1):71–109, 2003.

[20] Richard Fikes and Tom Kehler. The role of frame-based representation in reasoning. *Communications of the ACM*, 28(9):904–920, 1985.

[21] Michael R Genesereth and Nils J Nilsson. Logical foundations of artificial. *Intelligence. Morgan Kaufmann*, 58, 1987.

[22] Thomas R Gruber et al. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[23] AG Hamilton. Lógica para matemáticos, isbn 8428311013. *Editorial Paraninfo SA*, 1981.

[24] Frederick Hayes-Roth and Neil Jacobstein. The state of knowledge-based systems. *Communications of the ACM*, 37(3):26–39, 1994.

[25] Marcellin Jacques. Sergio. *Inteligencia Artificial, Aprendizaje y Sistemas Expertos. Maestría en Ciencias de la Computación, IIMAS-UNAM*, 1997.

[26] Guillermo Blasco Jiménez. Modelos gráficos probabilísticos en sistemas distribuidos.

[27] Dervis Karaboga. Artificial bee colony algorithm. *scholarpedia*, 5(3):6915, 2010.

[28] Patricia Kitcher. Marr's computational theory of vision. *Philosophy of Science*, 55(1):1–24, 1988.

[29] Sotiris Kotsiantis. A hybrid decision tree classifier. *Journal of Intelligent & Fuzzy Systems*, 26(1):327–336, 2014.

[30] Angel Fernando Kuri-Morales, Edwin Aldana-Bobadilla, and Ignacio López-Peña. The best genetic algorithm ii. In *Mexican International Conference on Artificial Intelligence*, pages 16–29. Springer, 2013.

[31] Alberto Colorni Marco Dorigo Vittorio Maniezzo. Distributed optimization by ant colonies. In *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, page 134. Mit Press, 1992.

[32] Sergio Marcellin. El k=conocimiento. *Notas del Curso Sistemas Expertos*, 2006.

[33] Sergio Marcellin. La representación del conocimiento en el desarrollo de los sistemas expertos. *Notas del Curso Sistemas Expertos*, 2006.

[34] Nax P Mendler, Prakash Panangaden, Philip J Scott, and RAG Seely. A logical view of concurrent constraint programming. *Nord. J. Comput.*, 2(2):181–220, 1995.

[35] Marvin Minsky. Minsky's frame system theory. In *TINLAP*, volume 75, pages 104–116, 1975.

[36] Eduardo Morales and L. Enrique Sucar. Manejo de incertidumbre. *Notas del Curso de Inteligencia Artificial del INAOEP*, 2012.

[37] Allen Newell. The knowledge level. *Artificial intelligence*, 18(1):87–127, 1982.

[38] Carl Adam Petri. Kommunikation mit automaten. 1962.

[39] Luis Rincón. Introducción a la probabilidad. 2013.

[40] Salvatore Ruggieri. Efficient c4. 5 [classification algorithm]. *IEEE transactions on knowledge and data engineering*, 14(2):438–444, 2002.

[41] Francisco Ruiz, Aurora Vizcaíno, Mario Piattini, and Félix García. An ontology for the management of software maintenance projects. *International Journal of Software Engineering and Knowledge Engineering*, 14(03):323–349, 2004.

[42] Enciclopedia Salvat. Enciclopedia salvat, 1974.

[43] F Santos. *Un Système Hybride Neuro-Symbolique pour l'Apprentissage Automatique Constructif*. PhD thesis, Tesis Doctoral. Laboratoire LEIBNIZ/INPG, Grenoble–França, 1998.

[44] Shiju Sathyadevan and Remya R Nair. Comparative analysis of decision tree algorithms: Id3, c4. 5 and random forest. In *Computational Intelligence in Data Mining-Volume 1*, pages 549–562. Springer, 2015.

[45] Néstor Manuel Garzón Torres and Luis Carlos Torres Soler. Ingeniería del conocimiento.

[46] William A Woods. What's in a link: Foundations for semantic networks. *Representation and understanding: Studies in cognitive science*, pages 35–82, 1975.