INFORME FINAL DEL TRABAJO DE GRADO

INTELIGENCIA ARTIFICIAL
APRENDIZAJE COMPUTACIONAL

DIRECTOR
Ana Monteiro

CO-DIRECTOR
Javier Díaz

AUTORES
Mónica Melendez
Roberto Potente

A nuestros padres

ÍNDICE

Introducción

Cuando comenzamos el estudio sobre aprendizaje computacional contábamos con investigaciones de diferentes autores, desconociendo si entre ellos existía algún tipo de relación.

Luego de leer y analizar cada uno de los papers, descubrimos que existían aspectos que podían ser comparados. Esto nos permitió ver la posibilidad de organizar esta información teniendo en cuenta dichos aspectos.

La clasificación elegida para organizar los papers es la clasificación es clasificación basada sobre la estrategia de aprendizaje subyacente, porque esta considera más aspectos de la información provista por una fuente, que es con lo único que cuenta inicialmente quien quiere aplicar aprendizaje.

Para cada estrategia de aprendizaje se describen métodos que la implementan, tratando de desarrollar la mayor cantidad posibles de ellos a fines de permitirnos realizar una buena comparación, perdiendo en cada descripción exaustividad.

A partir del análisis mencionado, no solo creímos útil organizar la información dispersa, sino también desarrollar una serie de comparaciones que describan diferencias y similitudes entre las distintas formas de desarrollar aprendizaje computacional, considerando esto un aporte necesario.

El resultado del análisis descripto anteriormente dio origen al survey de aprendizaje computacional, cuya utilidad es asistir a quien necesita aplicar este tipo de aprendizaje en un dominio particular.

Frente a un problema real, el survey intente ayudar a descubrir cual es el método más apropiado para solucionarlo: a través de la comparación entre estrategias que profundizan determinadas características de los problemas que solucionan cada una, se intenta encontrar la más adecuada para resolver dicho problema.

Una vez seleccionada una estrategia, se procederá de la misma manera a través de la comparación de métodos que implementan dicha estrategia para seleccionar uno en particular. Este procedimiento no siempre dará como resultado una única alternativa.

Para testear la utilidad de survey para solucionar un problema de aprendizaje computacional consideramos necesario aplicarlo sobre un dominio real, concluyendo con un prototipo que demostrará la factibilidad práctica.

PARTE I

Survey de Aprendizaje Computacional

1

Aprendizaje computacional

1.1) Introducción

El aprendizaje es un fenómeno multifacético, que incluye la adquisición de nuevo conocimiento, el desarrollo de habilidades cognitivas y motoras a través de instrucción o práctica, la organización de nuevo conocimiento en uno más general, la construcción de representaciones efectivas y el descubrimiento de nuevos hechos y teorías a través de observaciones y experimentaciones. Tales capacidades pueden ser divididas en 2 formas básicas de aprendizaje:

1.- Adquisición de conocimiento: es un proceso definido como el aprendizaje de nuevas estructuras simbólicas de conocimiento y modelos mentales junto a la habilidad de aplicar esa información de una manera efectiva.

2.- Refinamiento de habilidades: es un proceso subconsciente por virtud de la repetición práctica, coordinación mental y motora y la corrección de desviaciones. En esta forma de aprendizaje, existe una fase inicial para adquirir conocimiento sobre como realizar la actividad.

Aprendizaje es un término muy amplio; esto lleva a que existan varias definiciones según diferentes puntos de vista. Según Simon [5]:

> "Aprendizaje denota cambios en sistemas que son "adaptables", en el sentido que ellos permiten que el sistema haga la misma tarea o tareas, en el mismo contexto, más eficientemente las próximas veces"

Minsky [12] reemplazó este criterio por uno más general:

"Aprendizaje es hacer cambios útiles en nuestra mente"

Otra forma de capturar los aspectos fundamentales de aprendizaje está dada por el hecho que él está involucrado con adquisición de conocimiento. Es obvio que para adquirir conocimiento es necesario representarlo. Esto lleva a la siguiente caracterización del aprendizaje [12]:

"Aprendizaje es construir o modificar la representación de lo que se está tomando experiencias"

Desde la aparición de las computadoras se trató de implantar capacidades de aprendizaje en un sistema computacional. Con la excepción de los programas experimentales desarrollados en investigaciones de aprendizaje computacional, los sistemas carecen de la habilidad para adquirir o generar conocimiento a partir del que ya tienen. Esto contrasta con la inteligencia humana, la cual tiene la habilidad de adquirir nuevo conocimiento y nuevas habilidades y mejorar con la práctica.

Como la habilidad de aprender y el comportamiento inteligente están vinculados, un objetivo de investigación en inteligencia artificial es entender la naturaleza del aprendizaje e implementar capacidades de aprendizaje en computadoras.

Surge luego la pregunta: ¿tal objetivo es deseable y alcanzable?.

La alcanzabilidad está dada por la respuesta a la siguiente pregunta: ¿Se pueden identificar criterios generales tal que si son satisfechos por una computadora, podríamos decir que ella "aprende"?. En algunos programas se pueden identificar ciertas habilidades que podrían insinuar que ellos aprenden, lo que no se sabe es hasta donde se va a poder progresar usando el hardware de computadoras convencional y los métodos de programación existentes.

Es deseable desarrollar sistemas de aprendizaje computacional porque son necesarios para asegurar el progreso en inteligencia artificial. Esto se ha visto en varias áreas: sistemas expertos o sistemas de gran escala, sistemas basados en conocimiento, entendimiento de lenguaje, sistemas tutoriales inteligentes e interfaces amigables hombre-máquina.

Simon [5] compara permanentemente aprendizaje computacional con aprendizaje humano y todo lo que el involucra (la innata capacidad de adquirir hechos, habilidades y conceptos más abstractos); pero según otros investigadores [17] no hay razón para creer que esa es la única forma de adquirir hechos y habilidades, sino que es sólo un punto en el espacio de posibles métodos de aprendizaje.

1.2) Clasificación de aprendizaje computacional

Según [17] y [12], se puede clasificar a los sistemas de aprendizaje computacional en 5 dimensiones:

A.- Clasificación basada sobre la estrategia de aprendizaje subyacente.

En cualquier situación de aprendizaje, el sistema transforma la información provista por una fuente en alguna forma nueva, la cual es almacenada para su futuro uso. La naturaleza de esta transformación determina el tipo de estrategia usada. La siguiente clasificación está ordenada por la complejidad de dicha transformación:

A1.- Aprendizaje de rutina e incorporación directa de nuevo conocimiento: no se realiza ningún tipo de inferencia; existen 2 tipos: por programación y por memorización de hechos y datos.

A2.- Aprendizaje a partir de instrucción: el sistema selecciona y reformula el conocimiento dado por una fuente en una representación de conocimiento interna.

A3.- Aprendizaje por analogía: es adquirir nuevos hechos o habilidades transformando y aumentando conocimiento existente que tenga similitud con el nuevo concepto o habilidad que se desea resolver más eficientemente.

A4.- Aprendizaje a partir de ejemplos: dado un conjunto de ejemplos, el sistema determina una descripción de concepto general que cubre todos los ejemplos positivos y ninguno de los negativos (véase 2.1).

A5.- Aprendizaje a partir de observación y descubrimiento: busca sin ayuda de un instructor, regularidades y reglas generales explicando todas o algunas de las observaciones.

B.- Clasificación de acuerdo al tipo de conocimiento adquirido:

Un sistema de aprendizaje puede adquirir distintos tipos de conocimiento, dependiendo principalmente de la representación utilizada.

B1.- Parámetros en expresiones algebraicas: consiste en ajustar parámetros para obtener la performance deseada.

B2.- Arboles de decisión: sirve para discriminar entre clases de objetos.

B3.- Gramáticas formales: se utilizan para reconocer un lenguaje particular (usualmente artificial).

B4.- Reglas de producción: son pares **Condición => Acción** simples y fáciles de interpretar.

B5.- Expresiones basadas en lógica formal y formalismos relacionados: son expresiones usadas para describir los objetos individuales de entrada y las descripciones de conceptos resultantes.

B6.- Grafos y redes: proveen una representación más eficiente y conveniente que las expresiones lógicas.

B7.- Frames y esquemas: proveen una unidad de representación más extensa que las reglas de producción y las expresiones lógicas. Por ejemplo, un sistema que como unidad de representación utilice planes generalizados.

B8.- Programas de computadoras y otras codificaciones procedurales: son sistemas cuyo objetivo es adquirir una habilidad para llevar a cabo un proceso eficientemente en vez de detenerse en sus pasos internos.

B9.- Taxonomías: las taxonomías o jerarquías de los objetos del dominio son el resultado del aprendizaje a partir de observaciones.

B10.- Múltiples representaciones: algunos sistemas de adquisición de conocimiento usan varios esquemas de representación para el nuevo conocimiento.

C.- Clasificación por el dominio de aplicación: agricultura, química, música, etc.

D.- Clasificación de acuerdo al paradigma de investigación:

Estas aproximaciones difieren principalmente en la cantidad de conocimiento a priori incorporado al sistema de aprendizaje y la manera en que el conocimiento es representado y modificado:

D1.- Técnicas de modelización neuronal y decisiones teóricas: se esfuerza por desarrollar sistemas de aprendizaje de propósito general, partiendo de poco conocimiento inicial y usando parámetros continuamente cambiables. En esta aproximación los algoritmos y métodos de aprendizaje son de carácter numérico.

D2.- Adquisición de conceptos simbólicos (SCA): el sistema aprende construyendo una representación simbólica de un conjunto de conceptos, dados a través de

ejemplos y contraejemplos de ellos. Los atributos y predicados relevantes al concepto son provistos al sistema por el instructor.

D3.- Aprendizaje en un dominio específico con conocimiento intensivo (KDL): el sistema contiene numerosos conceptos predefinidos, estructuras de conocimiento, restricciones de dominio, reglas heurísticas y transformaciones incorporadas relevante al dominio específico para el cual es sistema es construido. No todos los atributos o conceptos relevantes son provistos inicialmente, es esperado que el sistema derive nuevos atributos y conceptos en el proceso de aprendizaje.

D4.- Múltiples paradigmas: algunos sistemas desarrollados presentan una mezcla de las aproximaciones anteriores; una interesante combinación de SCA y KDL es un sistema basado en la idea de "módulos de conocimiento intercambiables".

E.- Clasificación de acuerdo a la orientación del aprendizaje:

E1.- Análisis teórico y desarrollo de algoritmos de aprendizaje generales: es la exploración teórica del espacio de los posibles algoritmos y métodos de aprendizaje, independientemente del dominio.

E2.- El desarrollo de modelos computacionales de procesos de aprendizaje humano: el enfoque es el aprendizaje humano y el objetivo es el desarrollo de teorías computacionales y modelos experimentales de aprendizaje humano.

E3.- Estudios orientados a tareas: se refiere a la construcción de sistemas de aprendizaje para aplicaciones específicas. Se desarrolla y analiza sistemas de aprendizaje para mejorar la performance de un conjunto predeterminado de tareas.

1.3) Paradigma inductivo

El survey desarrollado tiene en cuenta la clasificación basada en la estrategia de aprendizaje subyacente[1] . En general, estas estrategias se basan en el paradigma inductivo.

[1] Los métodos que apliquen más de una estrategia serán desarrollados en la clasificación correspondiente de acuerdo a la estrategia más significativa.

Según [10], la habilidad de las personas para realizar generalizaciones o descubrir patrones a partir de observaciones aparentemente caóticas, es logrado a través del aprendizaje inductivo: inferencia inductiva desde hechos provistos por un instructor o por un experto.

El objetivo de la inferencia inductiva es formular afirmaciones admisibles (descripciones correctas y completas) que expliquen los hechos dados y sean capaces de predecir nuevos hechos. Pero existe una debilidad, y es que el conocimiento adquirido no puede ser validado. Esto sucede porque las inferencias inductivas son inferencias que no "preservan verdad"; como consecuencia de esto el conocimiento adquirido es conjetural, o sea que aunque el sistema inductivo se le den entradas correctas, eso no garantiza que las salidas lo sean. En sistemas deductivos las inferencias "preservan verdad".

Los métodos inductivos de acuerdo a la estrategia de control se dividen en bottom-up y top-down [6]:

- Bottom-up (manejando datos): a partir de los datos de entrada se intenta gradualmente evolucionar hacia la solución del problema.
- Top-down (manejando modelos): busca el conjunto de posibles soluciones (a partir de un modelo) en un intento de encontrar las "mejores" hipótesis que satisfacen ciertos requerimientos.

Los métodos bottom-up tienden a ser más rápidos, pero se ven menos afectados por el ruido (ejemplos con errores) y son menos flexibles. En cambio los métodos top-down tienen buena inmunidad al ruido y pueden ser fácilmente modificados para descubrir otras formas de generalizaciones. Una desventaja de estos últimos es que ellos son computacionalmente costosos y deben testear cada hipótesis para verificar si cubren los ejemplos de entrada.

El tipo más común de aprendizaje inductivo es el aprendizaje inductivo conceptual, que intenta aprender descripciones simbólicas, expresadas en términos orientados a humanos, aplicadas a objetos o fenómenos del mundo real.

Los dos tipos más comunes de aprendizaje inductivo conceptual son: **aprendizaje conceptual a partir de ejemplos** (también llamado adquisición de conceptos) y **aprendizaje conceptual a partir de observaciones** (también llamado generalización descriptiva).

El lenguaje de representación debe ser elegido de tal manera que las características fundamentales puedan ser fácilmente codificadas y que la información irrelevante pueda ser fácilmente ignorada.

Dado un conjunto de hechos observados se puede construir un número potencialmente infinito de afirmaciones inductivas, por lo cual es necesario incluir información adicional para

restringir el espacio de posibles afirmaciones inductivas y localizar las más deseables. Por lo tanto, un aspecto importante a tener en cuenta en el paradigma inductivo es el **sesgo** (en [9]): cualquier factor que influya sobre la selección de una afirmación inductiva (o hipótesis), excepto los hechos observados. Estos factores incluyen:

1.- El lenguaje en el cual las hipótesis son descriptas.

2.- El espacio de hipótesis que el programa puede considerar.

3.- Los procedimientos que definen en que orden las hipótesis serán consideradas.

4.- El criterio de aceptabilidad que define si un procedimiento de búsqueda puede parar cuando se confirma una hipótesis dada o debe continuar buscando una mejor.

Se dice que un sesgo es fuerte si dirige al sistema de aprendizaje de conceptos a un número pequeño de hipótesis; opuestamente opera un sesgo débil.

Se dice que un sesgo es correcto cuando permite al sistema de aprendizaje de conceptos seleccionar el concepto objetivo; caso contrario el sesgo de denomina incorrecto.

Cuando se tiene una sesgo fuerte y correcto la tarea del sistema es simple, porque rápidamente seleccionará el concepto objetivo; en cambio cuando se tiene un sesgo débil no se puede hacer mucho más que una selección al azar.

Por último el paradigma general para la inferencia inductiva es:

Dado:

a) hechos observados.

b) una afirmación inductiva tentativa (puede ser nula).

c) conocimiento del ambiente (incluyendo el sesgo).

Se encuentra:

Una afirmación inductiva, que implique los hechos observados y satisfaga el conocimiento del ambiente.

2

Aprendizaje a partir de ejemplos

2.1) Introducción

Aprendizaje a partir de ejemplos es un caso especial de aprendizaje inductivo. Este tipo de aprendizaje induce una descripción de un concepto general a partir de instancias específicas de él, llamadas ejemplos positivos, y de instancias de otros conceptos, llamadas ejemplos negativos o contraejemplos. Esta estrategia es usada también para aprendizaje de múltiples conceptos.

Este tipo de aprendizaje puede ser de un paso, donde todos los ejemplos son presentados a la vez, o incremental, donde el sistema forma una hipótesis del concepto consistente con los datos disponibles y subsiguientemente refina la hipótesis considerando ejemplos adicionales.

Algunos de los usos de aprendizaje a partir de ejemplos son [10]:

1) Aprendizaje de descripciones característica de una clase de objetos : especifica todas las propiedades comunes de los objetos conocidos en una clase. De esta manera define la clase en el contexto de un número ilimitado de otras clases.
2) Aprendizaje de descripciones discriminantes de una clase de objetos: especifica las propiedades de los objetos de la clase, que los distingue de los objetos en un número limitado de otras clases.
3) Inferir reglas de extrapolación de secuencias: predice el próximo elemento en una secuencia.

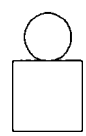
2.2) Métodos implementados

A continuación se describen siete métodos correspondiente a este tipo de aprendizaje, intentando mostrar su utilidad en diferentes situaciones.

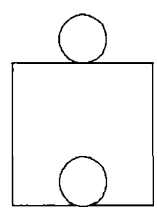
2.2.1) SPROUTER

El método SPROUTER (Hayes-Roth [6]), obtiene **generalizaciones conjuntivas máximamente específicas**[2], a partir de ejemplos positivos. Los eventos de entrada y sus generalizaciones están representados en representación estructural parametrizada (PSR). Tal representación es una conjunción de "CASE FRAME", estos están compuestos por "CASE LABELS" y parámetros; por ejemplo:

Evento 1: {{circulo: a},{cuadrado: b},{pequeño: a},{pequeño: b}, {arriba: a, b}}



Evento 2: {{circulo: c},{cuadrado: d},{circulo: e},{pequeño: c},{grande: d},{pequeño: e},
{arriba: c,d},{adentro: e,d}}



Este modelo es bottom-up. La primera generalización es el primer ejemplo (Evento 1). Dado un conjunto de generalizaciones y un ejemplo, realizando confrontaciones parciales, se obtiene un próximo conjunto de generalizaciones. Este nuevo conjunto de generalizaciones es obtenido en 2 pasos: el primer paso consiste en encontrar un conjunto M, de todas las posibles formas de confrontación entre un "CASE FRAME" del ejemplo y uno del conjunto de generalizaciones.

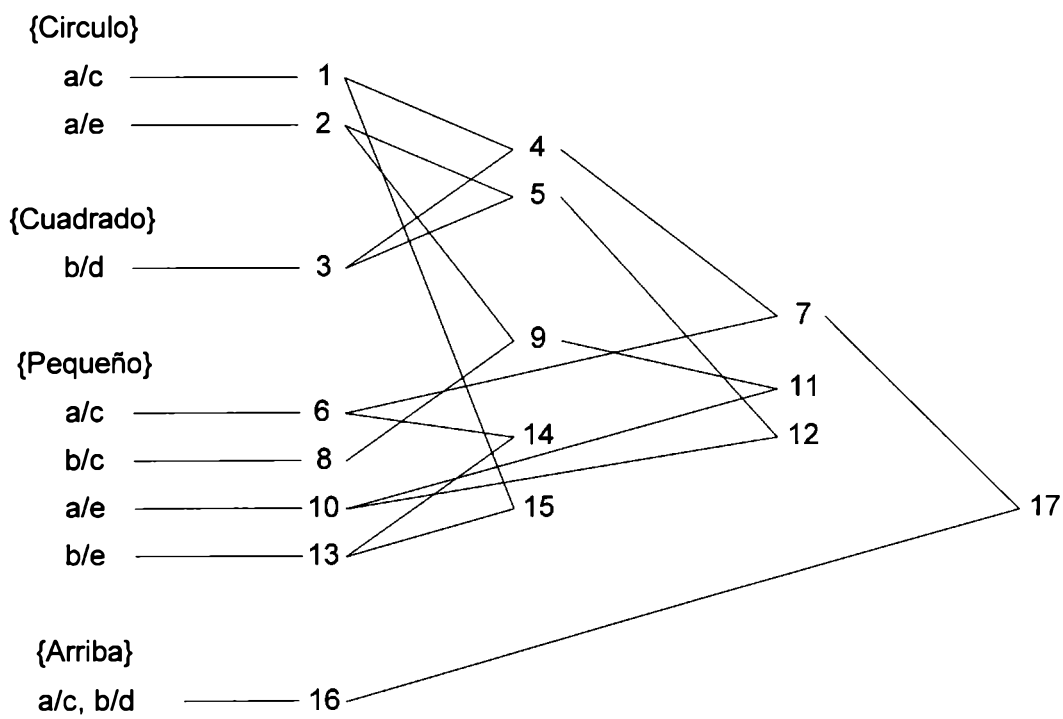
---

[2] Una generalización conjuntiva es una descripción de una clase de objetos formada por la conjunción de un grupo de sentencias primitivas. Una generalización conjuntiva maximamente especifica es la descripción más detallada (más específica) que describe todos los objetos conocidos de la clase.

Dos CASE FRAME confrontan si todos sus CASE LABELS son iguales. Cada elemento del conjunto M es un CASE FRAME y una lista de correspondencia de parámetros. En el ejemplo, del primer conjunto de generalizaciones (Evento 1) y el Evento 2 surge el siguiente conjunto M:

M = {{circulo (a/c),(a/e)},{cuadrado (b/d)},{pequeño (a/c),(b/c),(a/e),(b/e)},{arriba (a/c b/d)}}

El segundo paso selecciona subconjuntos de correspondencia de parámetros de M, tal que todos los parámetros puedan ser unidos consistentemente. Esto puede ser visualizado como el proceso de construcción de un grafo: se toma un elemento de M que constituye el primer nodo del grafo, sucesivamente se toma otro elemento de M, se lo incorpora al grafo como nodo, y se lo compara con todos los nodos anteriomente creados. Cuando este nuevo nodo es consistente con un nodo anterior (cada parámetro se corresponde de una única manera) se genera un nuevo nodo:



Con los nodos terminales del grafo se obtienen las generalizaciones conjuntivas máximamente específicas. Por ejemplo del nodo 17 se forma:

{{circulo: v1},{pequeño: v1},{cuadrado: v2},{arriba: v1,v2}}

2.2.2) Thoth

Este método (Vere [6]), obtiene generalizaciones conjuntivas máximamente específicas a partir de ejemplos positivos. Los ejemplos y las generalizaciones están representadas por literales (lista de constantes llamadas términos entre paréntesis); por ejemplo, considerando los mismos eventos del método anterior:

Evento 1: (circulo a)(cuadrado b)(pequeño a)(pequeño b)(arriba a b)

Evento 2: (circulo c)(cuadrado d)(circulo e)(pequeño c)(grande d)(pequeño e)
(arriba c d)(adentro e d)

Esta representación no tiene estructura lógica, trata a todos los términos uniformemente.

Este método es bottom-up; procesa los ejemplos uno a la vez para obtener el conjunto de generalizaciones conjuntivas.

El primer conjunto de generalizaciones es el primer ejemplo de entrada (Evento 1). Dado un conjunto de generalizaciones y un nuevo ejemplo, el algoritmo de generalización realiza los siguientes 4 pasos.

El primer paso es formar un conjunto MP, con todas las formas posibles de confrontar un literal del ejemplo con un literal del conjunto de generalizaciones. Dos literales confrontan si tienen el mismo número de términos y al menos un término común en una posición *común*. El MP resultante es:

MP = { ((circulo a),(circulo c)), ((circulo a),(circulo e)), ((cuadrado b),(cuadrado d)),
((pequeño a),(pequeño c)), ((pequeño a),(pequeño e)),
((pequeño b),(pequeño c)), (pequeño b),(pequeño e)), ((arriba a b),(arriba c d)) }

El segundo paso selecciona todos los posibles subconjuntos de MP tal que ningún literal confronte en más de una forma. Por ejemplo dos de ellos son:

MP1 = { ((circulo a),(circulo c)), ((cuadrado b),(cuadrado d)), ((pequeño a),(pequeño c)),
((arriba a b),(arriba c d)) }

MP2 = { ((circulo a),(circulo e)), ((cuadrado b),(cuadrado d)), ((pequeño a),(pequeño c)),
((arriba a b),(arriba c d)) }

En el tercer paso, a cada uno de los subconjuntos seleccionados en el paso 2, se les agrega pares de literales adicionales que no confrontaron previamente. Un par es agregado a un subconjunto si él está relacionado con otro par de ese subconjunto. Por ejemplo a MP1 se le puede agregar el siguiente par de literales:

((arriba a b),(adentro e d))

por estar relacionado con ((cuadrado b),(cuadrado d)).

En el último paso, de cada subconjunto de MP modificado (paso 3) se obtiene una generalización. Cada una es generada convirtiendo los pares de literales en una conjunción de literales, donde cada par forma un elemento de la conjunción. Por ejemplo la generalización obtenida a partir de MP1 es:

Gen 1: (circulo v1)(cuadrado v2)(pequeño v1)(arriba v1 v2) (v3 v4 v2)

2.2.3) Induce

Según Michalski y Buchanan [6], este método busca generalizaciones conjuntivas máximamente específicas. Los eventos de entrada y las generalizaciones están en lenguaje VL_{21}[3], donde cada evento es una conjunción de selectores. Un selector es una sentencia relacional que contiene un descriptor, con objetos o variables como argumento, y, si él representa una función, la lista de valores que puede tomar; por ejemplo, considerando los mismos eventos del método anterior:

Evento 1 = ∃ v1,v2 [tamaño(v1) = pequeño] [tamaño(v2) = pequeño] [forma(v1) = circulo][
forma(v2) = cuadrado] [arriba(v1,v2)]

Evento 2 = ∃ v1,v2,v3 [tamaño(v1) = pequeño] [tamaño(v2) = grande]
[tamaño(v3) = pequeño] [forma(v1) = circulo] [forma(v2) = cuadrado]
[forma(v3) = circulo] [arriba(v1,v2)] [adentro(v1,v2)]

El método utiliza una estrategia top-down.

---

[3] VL_{21} es una extensión de la lógica de primer orden que incorpora el concepto de selector, cuantificadores numéricos,etc.

Se intenta acelerar la búsqueda de generalizaciones admisibles, dividiendo la búsqueda en 2 pasos. El primer paso busca generalizaciones admisibles en el espacio de solo estructura (espacio definido por los descriptores que especifican estructura, que son los descriptores no unarios). Los descriptores unarios son removidos de los eventos de entrada para abstraerse al espacio de solo estructura. Los eventos resultantes son:

Evento 1' = ∃ v1,v2 [arriba(v1,v2)]

Evento 2' = ∃ v1,v2,v3 [arriba(v1,v2)] [adentro(v1,v2)]

Como resultado de este paso se obtiene un conjunto de generalizaciones candidatas: se forma un primer conjunto de generalizaciones con una muestra de eventos elegidos al azar. La generalizaciones de este conjunto que cubran todos los eventos de entrada, son pasadas a un conjunto de generalizaciones candidatas. Se forma un nuevo conjunto cuyos elementos son el resultado de generalizar los elementos restantes, eliminando un selector de todas las formas posibles. Así hasta que la cantidad de generalizaciones candidatas alcance un máximo preespecificado. En el caso que el conjunto de generalizaciones candidatas sea vacío, se elige otra muestra de eventos y se repite el proceso. El conjunto de generalizaciones candidatas en el ejemplo es:

{ ∃ v1,v2 [arriba(v1,v2)] }

En el segundo paso, todos los eventos de entrada son trasladados al espacio de solo atributos (espacio definido por los descriptores que especifican atributos, es decir los descriptores unarios), usando las generalizaciones candidatas del paso anterior: un evento trasladado es representado por una tupla de valores que resultan de confrontar el evento original con una generalización candidata. En el ejemplo los eventos trasladados son (de acuerdo al orden tamaño v1, tamaño v2, forma v1, forma v2):

Evento 1" = < pequeño, pequeño, circulo, cuadrado >

Evento 2" = < pequeño, grande, circulo, cuadrado >

Se buscarán las generalizaciones candidatas en el espacio de solo atributo usando los eventos trasladados como eventos de entrada. La búsqueda de generalizaciones candidatas en este espacio es similar a la búsqueda en el espacio anterior, sólo que, en vez de generalizar los elementos de un conjunto de generalizaciones, para formar el siguiente, eliminando

selectores, se generaliza extendiendo las disyunciones internas. Por lo tanto, las generalizaciones en este espacio están representadas por tuplas de disyunciones de valores. En el ejemplo, la generalización candidata en este espacio es:

< pequeño, pequeño v grande, circulo, cuadrado >

Combinando los descriptores que especifican atributos obtenidos en el segundo paso, con los descriptores que especifican estructura obtenidos en el primer paso, se construyen las generalizaciones conjuntivas de salida. En el ejemplo:

Gen = ∃ v1,v2 [tamaño(v1) = pequeño] [tamaño(v2) = pequeño v grande]
[forma(v1) = circulo] [forma(v2) = cuadrado] [arriba(v1,v2)]

La ventaja de este método es que la búsqueda en el espacio total es computacionalmente más cara que realizar la búsqueda en el espacio de solo estructura y luego la búsqueda en el espacio de solo atributo.

Una desventaja es que si nuestro problema usa sólo descriptores unarios o sólo descriptores no unarios, no existe ninguna ventaja computacional.

El usuario puede emplear varios criterios para evaluar cuán prometedora es una generalización intermedia. Estos criterios son combinados en una "evaluación funcional lexicográfica"[4] .

2.2.4) MACBETH

MACBETH [13] es un sistema que combina dos estrategias de aprendizaje: soluciona problemas por analogía y adquiere reglas a partir de ejemplos.

Este sistema está basado en una teoría que trabaja a partir de precedentes y ejercicios; los ejercicios son los problemas a solucionar y los precedentes son la base analógica para resolverlos. Por ejemplo, un instructor podría dar el siguiente precedente y ejercicio:

[4] Una evaluación funcional lexicográfica es una secuencia de pares criterios-tolerancia. Las generalizaciones son evaluadas por el primer criterio de la secuencia. Las que superan la tolerancia son evaluadas por el segundo criterio y así con todos los siguientes.

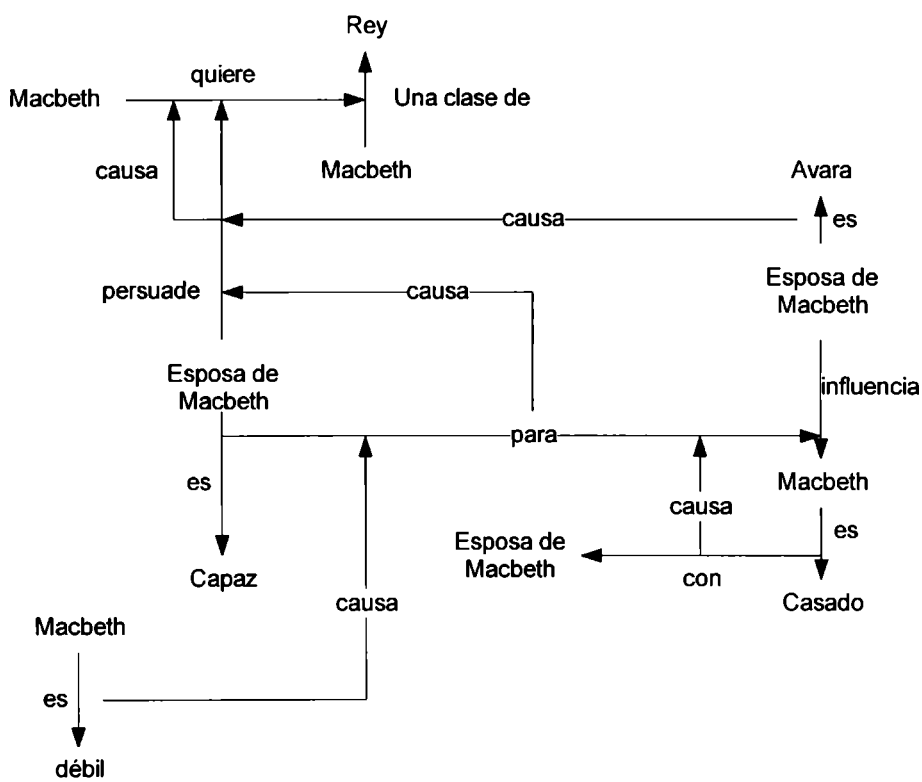
Precedente:

MA es una historia acerca de Macbeth, su esposa, Duncan, y Macduff. Macbeth es un noble malo, su esposa es una mujer avara y ambiciosa, Duncan es un rey, y Macduff es un noble. La esposa de Macbeth persuade a Macbeth a querer ser rey porque ella es avara. Ella es capaz de influenciarlo porque están casados y porque él es débil. Macbeth mata a Duncan con un cuchillo. Macbeth mata a Duncan porque él quiere ser rey y porque es malo. La esposa de Macbeth se suicida. Macduff está enojado. Macduff mata a Macbeth porque este mató a Duncan y porque Macduff es leal a Duncan.

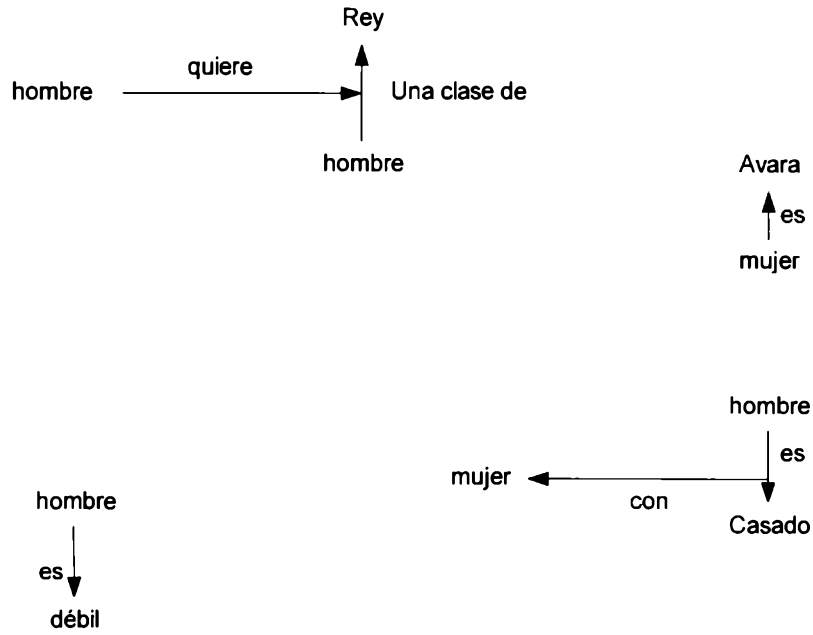
Ejercicio:

Sea E un ejercicio acerca de un noble débil y una señora avara. La señora está casada con el noble. En E mostrar que el noble puede querer ser rey.

El precedente y el ejercicio están representados por una "estructura causal": una red semántica, formada por partes y relaciones. El ejercicio carece de la unión entre las partes y relaciones conocidas y la relación a ser demostrada.



Estructura causal del precedente



Estructura causal del ejercicio

Dado un precedente y un ejercicio, los pasos para solucionarlo son:

. Las partes del precedente son contrastadas con las partes del ejercicio (por ejemplo: hombre con Macbeth).

. La estructura causal del precedente es mapeada sobre el ejercicio (Figura 4.1 y Figura 4.2). Si tal mapeo une las relaciones conocidas del ejercicio (por ejemplo: el noble es débil) con la relación a ser demostrada (por ejemplo: el noble quiere ser rey), el ejercicio es solucionado.

. Si el ejercicio fue solucionado, se construye una regla, donde el antecedente es una generalización de las relaciones conocidas en el ejercicio y la consecuencia es una generalización de la relación a ser demostrada.

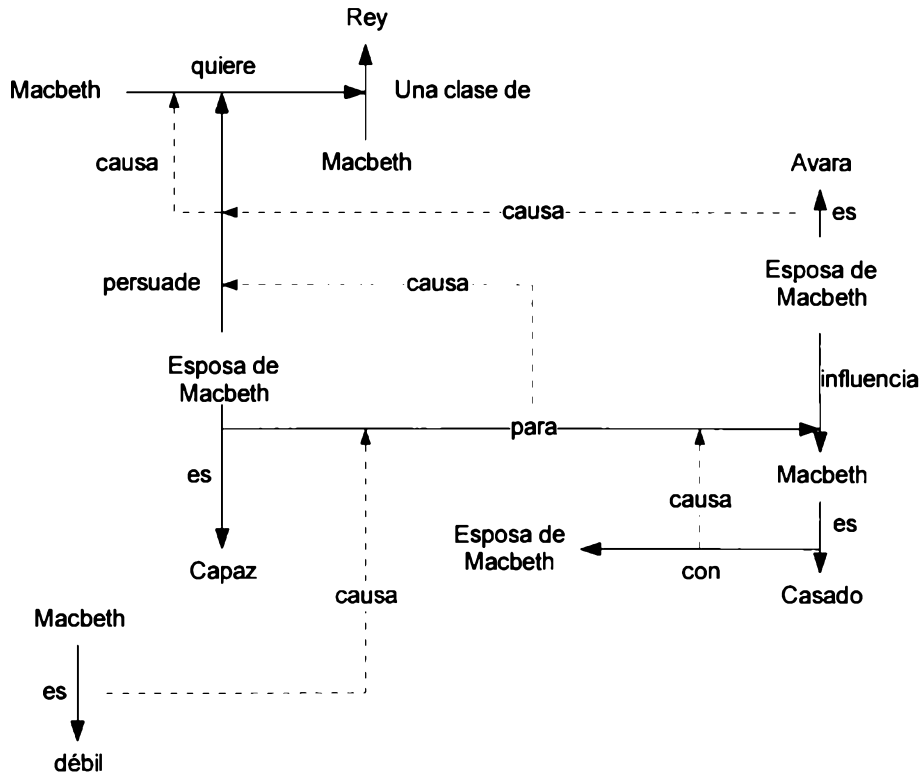


Figura 4.1

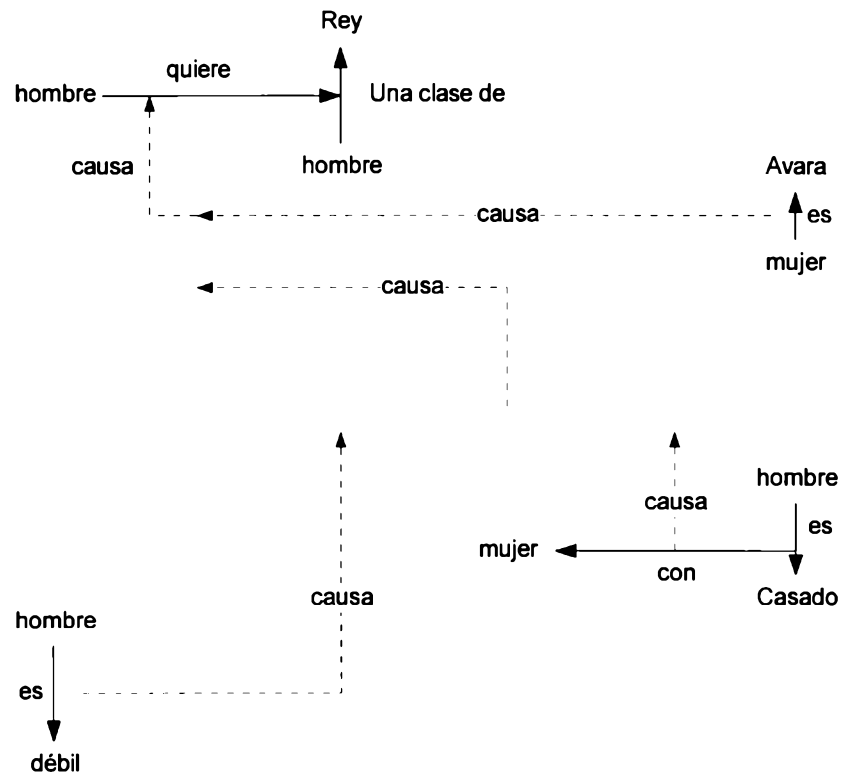


Figura 4.2

Como resultado la regla obtenida es:

REGLA-1 SI	Hay una señora avara y
	hay un noble débil y
	el noble está casado con la señora
ENTONCES	el noble puede querer ser rey.

Para evitar tener reglas demasiado generales, el sistema MACBETH extiende la teoría existente, basándose en 2 ideas:

1.- Principio de bloqueo: una regla no se aplicará a un dado problema, si alguna relación de la estructura causal del precedente, del cual se derivó la regla, es "manifiestamente inadmisible" en la nueva situación.
2.- Conjetura "prima-facie": una relación es manifiestamente inadmisible si su negación puede ser demostrada por un paso de inferencia directa, desde las relaciones existentes.

Por ejemplo dado un nuevo ejercicio:

> *Sea E un ejercicio acerca de un noble débil y una señora avara. La señora está casada con el noble. El no quiere a la señora. En E mostrar que el noble puede querer ser rey.*

la REGLA-1 es aplicable a este ejercicio, pero es demasiado general ya que se supone que una persona no puede influenciar a alguien si este no quiere a esa persona.

Para lograr el bloqueo de una regla, se extiende cada una con una parte "SALVO QUE". Cuando una regla es creada, las entradas a esta parte corresponden a las negaciones de todas las relaciones en la estructura causal del precedente, excepto aquellas con las que se formó el consecuente y la consecuencia. La REGLA-1 extendida es:

REGLA-1 SI	Hay una señora avara
	y hay un noble débil
	y el noble está casado con la señora
ENTONCES	el noble puede querer ser rey.
SALVO QUE	la mujer no persuade al noble a querer ser rey
	o la mujer no es capaz de influenciar al noble.

Una regla que es utilizada para bloquear a otra, es llamada "CENSOR". También un censor puede bloquear otro censor. Por ejemplo un censor que bloquea la REGLA-1 es:

CENSOR-1 SI una persona X no quiere a una persona Y.
ENTONCES Y no es capaz de influenciar a X.

El sistema toma la siguiente especialización de la conjetura "prima-facie": cuando una entrada en la parte "salvo que" corresponde a una relación que se puede demostrar que es verdadera, aplicando directamente otra regla a las relaciones existentes, sin más inferencias, la regla es bloqueada. Con esto, se quiere evitar hacer un profundo razonamiento (sucesivas inferencias) sobre las condiciones SALVO QUE.

Otras reglas que son usadas en una forma especial, además de los censores, son las "reglas de definición". Una regla de definición puede ser expresada por una regla extendida, esto permite que la parte "SALVO QUE" capture la idea de excepción a la definición.

Como vimos antes, las reglas extendidas, incluyendo los censores y las reglas de definición, pueden ser aprendidas usando precedentes y ejercicios. También se puede aprender a partir de instancias "near-misses"[5]. La idea de near-miss es útil en la situación en la que se ha adquirido una regla de definición incompleta. Cuando el instructor le indica al sistema que una instancia de una definición es incorrecta (instancia near-miss), el sistema calcula la diferencia entre la instancia y la definición, y la agrega como entrada en la parte salvo que de la regla de definición.

2.2.5) ID3

ID3 [8] es un mecanismo que a partir de ejemplos genera un árbol de decisión para clasificar objetos. Pertenece a la familia de sistemas de aprendizaje TDIDT (Top-Down Induction of Decision Trees) [7], donde los miembros de esta familia representan el conocimiento adquirido a través de árboles de decisión y utilizan una estrategia de aprendizaje no incremental.

---

[5] Una instacia near-miss es una instacia que difiere del concepto objetivo en solo algunas características significativas.

ID3 trabaja con ejemplos representados por atributos. La mayor atención estará centrada sobre la elección de esos atributos, para que ellos sean representativos y midan características importantes de los objetos.

Los objetos en el universo pertenecen a una clase de un conjunto de clases mutuamente exclusivas.

La regla de clasificación es construida a partir de un "training set" (subconjunto de objetos del universo de clase conocida). Se toma un subconjunto del training set (llamado window), elegido al azar, y con ello se forma un árbol de decisión; iterativamente se agrega a ese subconjunto "excepciones" (objetos que fueron mal clasificados por el árbol de decisión); y nuevamente se construye el árbol. Así hasta llegar a un árbol sin excepciones. Cuando se trabaja con muchos ejemplos, con esta estrategia se construyen árboles de decisión computacionalmente más rápidos que si se construye a partir de todo el training set.

El algoritmo para formar cada árbol de decisión es el siguiente: si todos los ejemplos no pertenecen a la misma clase, ID3 elige un atributo como raíz del árbol y crea tantos hijos como valores posibles toma ese atributo. En cada hijo se agruparán aquellos ejemplos cuyo valor del atributo corresponde al de esa rama. Se continúa de esta manera hasta llegar a nodos donde todos los ejemplos pertenecen a la misma clase. Allí, el conjunto de ejemplos es reemplazado por su clase.

El árbol de decisión debe capturar una relación significativa entre la clase de los objetos y los valores de los atributos, para poder clasificar correctamente no sólo los objetos del "training set", sino también otros objetos desconocidos en el momento de la formación del árbol.

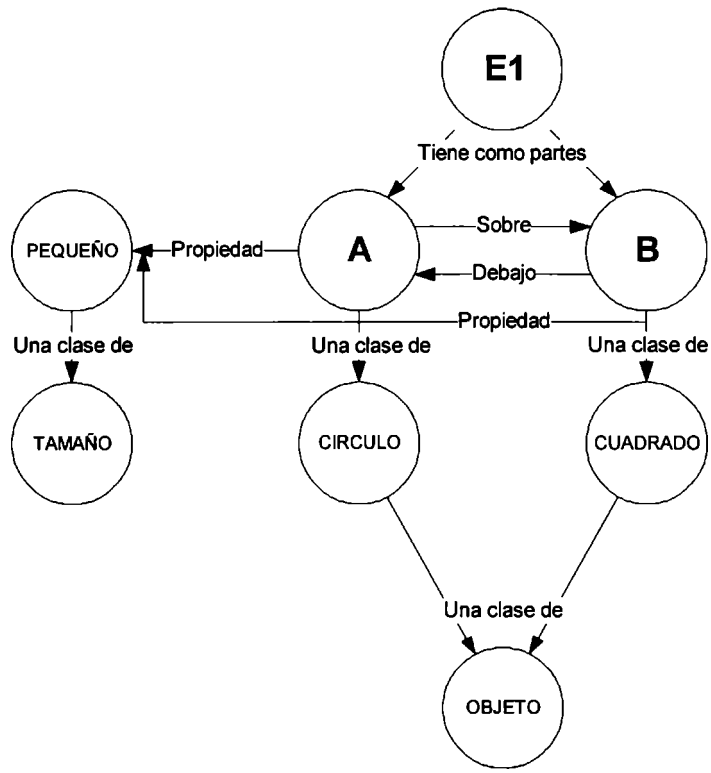
Una vez que se tiene el árbol de decisión, se clasifica un objeto de la siguiente manera: comenzando por la raíz del árbol, se testea el valor del atributo en el objeto, tomando la rama apropiada, así hasta llegar a una hoja.

La forma que tiene ID3 de elegir un atributo en cada paso de la formación del árbol, asegura que la cantidad de nodos del árbol final es mínima; con lo que en la clasificación de un objeto, la cantidad de test necesarios es mínima.

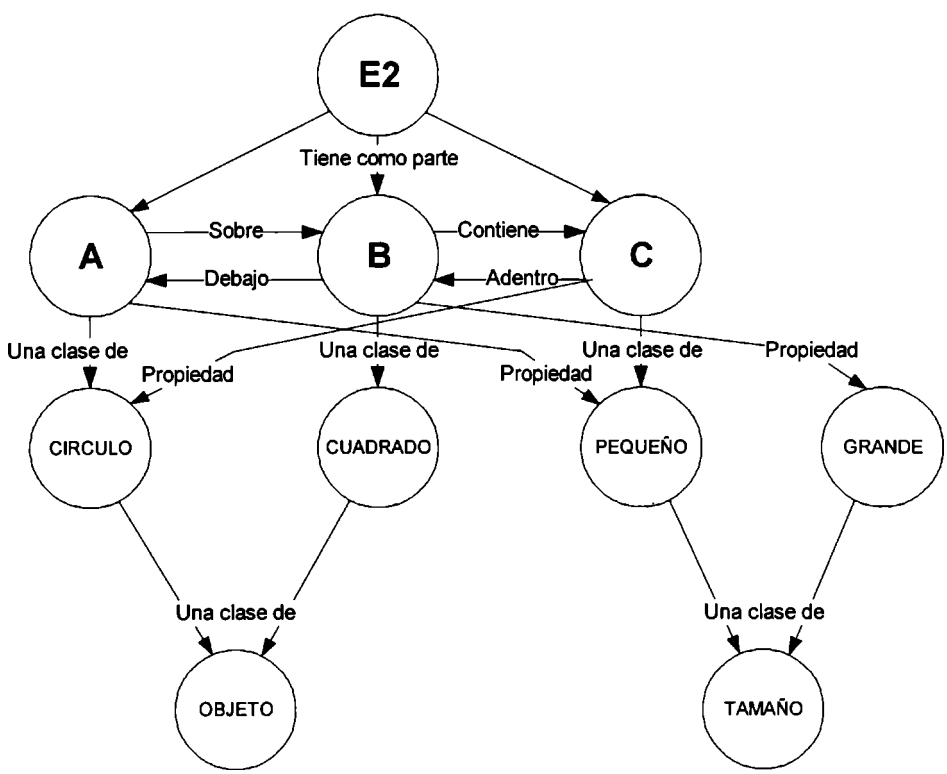
2.2.6) Aprendizaje de conceptos de bloques

Este método desarrollado por Winston [6], aprende conceptos que caracterizan bloques simples, en forma de generalizaciones conjuntivas máximamente específicas, a partir de ejemplos positivos y negativos.

Los ejemplos, el conocimiento del ambiente y las descripciones de conceptos generadas son representados por redes semánticas. Por ejemplo considerando los mismos eventos del método Sprouter:

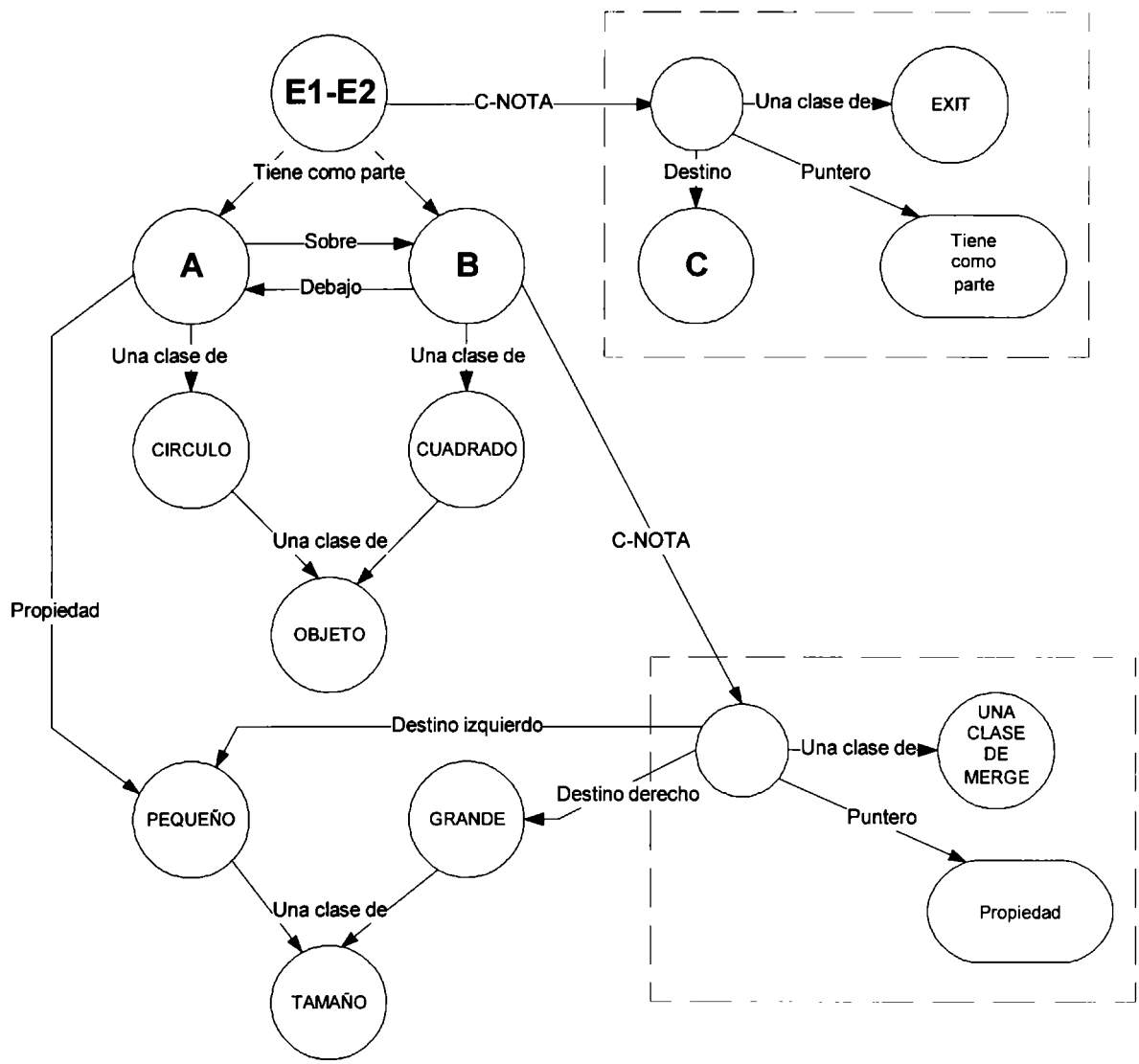


Evento 1

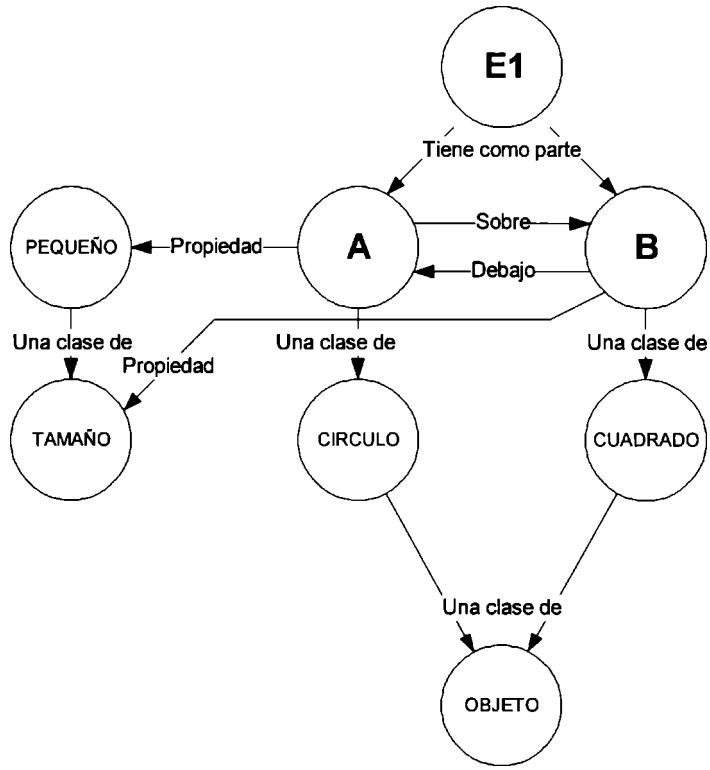


Evento 2

La estrategia seguida es bottom-up. La primera descripción de concepto es el primer ejemplo. Dada una descripción de concepto y un nuevo ejemplo, se obtendrá como primer paso una "descripción diferencia". Esta es obtenida realizando confrontaciones de grafos entre la descripción de concepto y el ejemplo dado, comentando estas confrontaciones con "C-NOTAS". Estas C-NOTAS son subredes que describen tipos de confrontaciones. Hay varios tipos de C-NOTAS de acuerdo a si la descripción del concepto y el ejemplo confrontan parcialmente o no confrontan. Lo obtenido es un "esqueleto generalizado" de lo que confronta exactamente, con las C-NOTAS acopladas. En el ejemplo, la descripción diferencia resultante es:



El segundo paso es obtener la generalización, a partir de la "descripción diferencia", manteniendo el esqueleto generalizado y manejando las C-NOTAS de acuerdo a su tipo y al tipo del ejemplo (positivo o negativo). La generalización resultante en el ejemplo es:



En algunos casos, como algunos tipos de C-NOTAS pueden ser manejadas de distintas formas, más de una generalización puede ser realizada. En tal situación se trabaja sobre una y se guardan las otras en una lista de "backtrack"[6] .

Winston utiliza condiciones "enfáticas" para dejar de lado los ejemplos negativos. Estas son condiciones del tipo "debe" o "no debe".

El usar "backtracking" y el construir las descripciones diferencias, en una situación del mundo real, puede llevar a problemas combinatorialmente infactibles.

2.2.7) Marvin

Marvin [16] es un programa que aprende conceptos complejos, usando los conceptos más simples aprendidos previamente (los cuales están almacenados en memoria), y haciendo

6 "Volver atras", para intentar con otra alternativa que fue dejada de lado anteriormente.

preguntas al usuario, es decir, combina aprendizaje a partir de instrucción con aprendizaje a partir de ejemplos.

Un concepto en Marvin es una descripción de un conjunto de objetos en un universo. El lenguaje de descripción de conceptos utilizado es un subconjunto de lógica primer orden. Un concepto es representado por un conjunto de cláusulas de Horn.

Marvin utiliza la forma de aprendizaje de "maestro a estudiante", primero debe aprender los conceptos más simple y luego los más complejos (ya que estos muchas veces hacen referencia a los primeros).

La búsqueda del concepto objetivo comienza cuando el usuario da un ejemplo positivo de él. Por ejemplo, para aprender el concepto de arco el usuario da el siguiente ejemplo:

A es_parte_de X
B es_parte_de X
C es_parte_de X
A está_sobre B
A está_sobre C
B está_sobre D
C está_sobre E
es_base(D)
es_base(E)
B está_a_la_izquierda_de C
B no_toca C
horizontal(A)
triángulo(A)
vertical(B)
rectángulo(B)
vertical(C)
rectángulo(C)

esto define la primera hipótesis (H0) del concepto a aprender. Luego Marvin encuentra que objetos en el ejemplo son instancias de los conceptos ya aprendidos, por ejemplo si se tiene rectángulo(B) en el ejemplo y la cláusula forma(X) <— rectángulo(X) en la memoria, Marvin puede generalizar el concepto reemplazando rectángulo(B) por forma(B). De esta manera a partir de los conceptos ya aprendidos forma una lista de reemplazos. Por ejemplo a partir de los conceptos forma, orientación, columna, etc., Marvin forma la siguiente lista de reemplazos:

forma(B) ← rectángulo(B)

forma(C) ← rectángulo(C)
forma(A) ← triángulo(A)

orientación(B) ← vertical(B)
orientación(C) ← vertical(C)
orientación(A) ← horizontal(A)

columna(B) ← rectángulo(B) & vertical(B) & B está_sobre D & base(D)
columna(C) ← rectángulo(C) & vertical(C) & C está_sobre E & base(E)

igual_altura (D,E) ← base(D) & base(E)
igual_altura (B,C) ← rectángulo(B) & vertical(B) & rectángulo(C) & vertical(C) &
B está_sobre D & C está_sobre E & igual_altura (D,E)

B adyacente_a C ← B está_a_la_izquierda_de C

B puede_tocar C ← B no_toca C

A puede_estar_sobre B ← A está_sobre B
A puede_estar_sobre C ← A está_sobre C
B puede_estar_sobre D ← B está_sobre D
C puede_estar_sobre E ← C está_sobre E

Marvin forma una hipótesis de concepto (H1) generalizando el ejemplo, con uno de los reemplazos de la lista encontrada. Por ejemplo, tomando el reemplazo forma(B) <— rectángulo(B):

A es_parte_de X
B es_parte_de X
C es_parte_de X
A está_sobre B
A está_sobre C
B está_sobre D
C está_sobre E
es_base(D)
es_base(E)
B está_a_la_izquierda_de C

B no_toca C
horizontal(A)
triángulo(A)
vertical(B)
vertical(C)
rectángulo(C)
forma (B)

Una vez hecha la hipótesis, construye un ejemplo, instancia de ella, y le pregunta al usuario si tal ejemplo pertenece al concepto objetivo. Esto lo hace para determinar si la hipótesis del concepto es consistente (o sea, cada objeto que pertenece a la hipótesis también pertenece al concepto objetivo) o no. Si es inconsistente significa que en la generalización se ha perdido información específica importante, por lo tanto Marvin especializa la hipótesis y vuelve a testear por consistencia. Para especializar debe ser agregada información adicional que califique la generalización; esto es hecho buscando otro reemplazo de la lista tal que la intersección entre su parte derecha y la parte derecha del reemplazo anterior no se vacía. Se aplica este reemplazo a la última hipótesis de concepto consistente (H0) y se une su resultado con la hipótesis inconsistente (H1).

En el ejemplo, si el usuario responde que la instancia de la hipótesis presentada no pertenece al concepto objetivo, Marvin especializa la hipótesis anterior utilizando el siguiente reemplazo:

columna(B) ← rectángulo(B) & vertical(B) & B está_sobre D & base(D)

obteniendo:

A es_parte_de X
B es_parte_de X
C es_parte_de X
A está_sobre B
A está_sobre C
C está_sobre E
es_base(E)
B está_a_la_izquierda_de C
B no_toca C
horizontal(A)
triángulo(A)

vertical(C)

rectángulo(C)

forma (B)

columna(B)

Notar que las cláusulas de Horn B está_sobre D, es_base(D) y vertical(B) no aparecen en la hipótesis anterior porque son subsumidas por columna(B).

Esto continua hasta que la hipótesis de concepto no pueda ser más generalizada. Entonces la descripción del concepto resultante es almacenada en memoria. La generalización encontrada para el ejemplo es:

A es_parte_de X

B es_parte_de X

C es_parte_de X

A está_sobre B

A está_sobre C

B no_toca C

columna(B)

igual_altura(B,C)

columna(C)

forma(A)

horizontal(A)

B adyacente_a C

Se pueden detectar errores en las descripciones de concepto, en ese caso se puede hacer una depuración siguiendo los pasos que Marvin realizó para aprenderla y pidiendo al usuario la confirmación de tales pasos.

2.3) Comparación y conclusión

A partir del análisis de los métodos descriptos anteriormente, creímos útil mostrar como cada uno de los métodos trata las características relevantes de aprendizaje a partir de ejemplos. La idea del cuadro es guiar al usuario de este survey a elegir el método que mejor se adecúe a sus necesidades.

* El trabajar sólo con ejemplos positivos puede llevar al problema de sobregeneralizar los ejemplos, por lo tanto, para evitarlo, es necesario introducir conocimiento del ambiente o heurísticas para acotar la generalización. Algunos ejemplos de esto son: el programa Thoth de Vere y el programa SPROUTER de Hayes_Roth.

  En cambio cuando se trabaja también con ejemplos negativos, ellos permiten dar un límite al nivel de generalización, por ejemplo el programa de Winston de aprendizaje de conceptos de bloques.

* Algunos de los sistemas que aplican estrategia de control bottom-up son el programa Thoth de Vere y el programa de Winston de aprendizaje de conceptos de bloques. En cambio el programa Induce de Michalsky aplica una estrategia de control top-down.

  Existen programas que utilizan un esquema mixto, top-down y bottom-up.

* Un método de aprendizaje que usa un lenguaje con poca estructura (tiene pocos operadores y pocos tipos de operandos) tiende a ser relativamente eficiente y fácil de implementar pero puede no ser capaz de aprender descripciones útiles en aplicaciones del mundo real. Por otro lado, un método que usa un lenguaje demasiado rico tendrá problemas de implementación.

  Como ejemplo de un lenguaje de poca estructura podemos nombrar la "estructura de lista no interpretada" utilizada por el programa Thoth, el cual no tiene ninguna estructura lógica. Como un ejemplo de lenguaje rico podemos mencionar a VL_{21}, una extensión de la lógica de primer orden utilizada por Induce. El tiene una gran variedad de formas representacionales, las que le permite describir cualquier problema específico. En un punto intermedio, podría estar ubicada la "representación estructural parametrizada" utilizada por el programa SPROUTER, que puede representar fácilmente predicados y relaciones.

  La necesidad de que un lenguaje de representación sea rico o no, depende de si el método es de propósito general u orientado a un problema. Por ejemplo, aunque las redes semánticas no están entre los lenguajes de representación más ricos, Winston, en su programa de aprendizaje de conceptos de bloques (método orientado a un problema) lo usa para poder representar cualquier problema en su dominio.

  La mayoría de los métodos nombrados hasta ahora, como casi todos los sistemas de IA, emplean un único modelo para representar las posibles descripciones finales.

* Cuando la fuente es el "instructor", él conoce el concepto a aprender, por lo tanto genera ejemplos significativos de él, como por ejemplo SPROUTER. La desventaja es

que posiblemente no conozca el estado del conocimiento del sistema; si lo conociera podría dar ejemplos de forma tal de poder llegar rápidamente al concepto deseado.

Si la fuente fuera el "sistema mismo", él conoce su estado de conocimiento, pero no sabe cual es el concepto a aprender. Dado un conjunto de posibles descripciones de conceptos, él generará sus ejemplos de manera tal que ellos permitan elegir algunas descripciones y descartar otras. Existe una entidad externa que clasifica dichos ejemplos. Por ejemplo, en el aprendizaje a partir de ejemplos del sistema Marvin, a partir de un primer ejemplo dado por un instructor, él genera sus propios ejemplos para aprender un concepto.

Si la fuente fuera el "ambiente externo", los ejemplos son dados al azar, y puede no tenerse en cuenta casos extremos e infrecuentes, por lo tanto estos no estarán incluidos en la descripción del concepto.

* Aprendizaje de múltiples conceptos vs aprendizaje de un único concepto: la mayoría de las investigaciones que han sido realizadas sobre aprendizaje de conceptos en IA, han consistido en darle al programa ejemplos (y posiblemente contraejemplos) de un concepto específico para que él determine su definición. Pero en situaciones del mundo real, una gran cantidad de ejemplos pueden definir varios conceptos, lo que implica aprendizaje de múltiples conceptos, por ejemplo ID3.

	Tipos de ejemplos	Estrategia de control	Lenguaje de represen-tación	Fuente de los ejemplos	Específicos o generales	Múltiples o único concepto
SPROUTER	Positivos	Bottom-up	Representa-ción estruc-tural para-metrizada	El instructor	General	Único concepto
THOTH	Positivos	Bottom-up	Estructura de lista no interpretada	El instructor	General	Único concepto
INDUCE	Positivos	Top-down	VL_{21} (una extensión de lógica de primer or-den)	El instructor	General	Único concepto

	Tipos de ejemplos	Estrategia de control	Lenguaje de representación	Fuente de los ejemplos	Específicos o generales	Múltiples o único concepto
MACBETH	Positivos y negativos a través de instancias near-miss	Bottom-up	Redes semánticas	El instructor	General	Único concepto
MARVIN	El usuario da positivos y Marvin genera positivos y negativos	Bottom-up	Cláusulas de Horn	El instructor y el sistema mismo	General	Único concepto
ID3	Los positivos de una clase actúan como negativos para las demás	Bottom-up	Arboles de decisión	El instructor	General	Múltiples conceptos disyuntos
Aprendizaje de conceptos de bloques	Positivos y negativos	Bottom-up	Redes semánticas	El instructor	Específico	Único concepto

3

Aprendizaje por analogía

3.1) Introducción

El mecanismo de aprendizaje por analogía trata de solucionar un problema basándose en experiencias pasadas: situaciones en las que se resolvieron problemas con características similares al problema que se quiere resolver.

Definición: solucionar problemas por analogía consiste en transferir conocimiento desde episodios pasados a nuevos problemas que comparten aspectos significativos con las experiencias pasadas correspondientes y usar el conocimiento transferido para construir soluciones para el nuevo problema [1].

Se puede justificar el aprendizaje por analogía asumiendo que, dado un problema, generalmente se tiene una situación similar y familiar que sirva como base para solucionarlo.

3.2) Analogía transformacional y derivacional

Existen dos formas de resolver problemas por analogía: analogía transformacional y analogía derivacional. Se van a comparar estas dos formas teniendo en cuenta los siguientes criterios:

. ¿Qué significa que los problemas comparten aspectos significativos?

. ¿Qué conocimiento es transferido desde una experiencia pasada a la nueva situación?

. ¿Cómo es transferido el conocimiento?

. ¿Cómo son seleccionadas las experiencias relacionadas analógicamente, desde la memoria de episodios para solucionar problemas?

En **analogía transformacional** [1] [11] se toma la solución de un problema similar al dado, y se la modifica de manera tal que sea solución del nuevo problema (Figura 1). Una solución está dada por una secuencia de acciones que si se aplican a un estado inicial producen el estado objetivo.

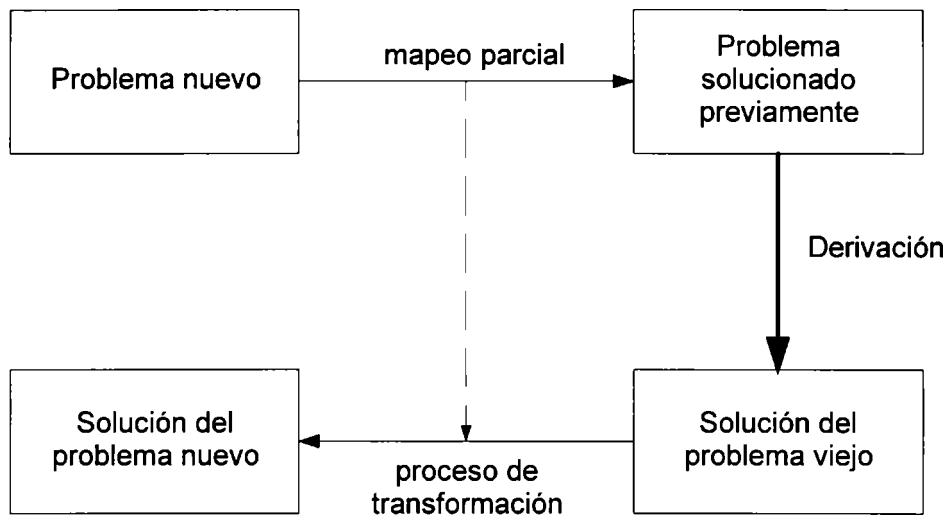


Figura 1: El proceso de analogía transformacional.

El proceso de transformación de soluciones es realizado por medio de un conjunto de operadores de transformación atómicos y la aplicación del método "means ends"[7] para solucionar problemas. Este proceso puede ser aplicado a un espacio de posibles soluciones (llamado T-espacio) para solucionar problemas por analogía.

El primer paso es "recordar" un problema previamente solucionado, cuya solución pueda transferirse al problema en consideración. Estos problemas previamente solucionados con sus correspondientes soluciones están almacenados en una memoria de episodios. Una "métrica de similitud" permite recuperar la solución de un problema previo, similar al presente; esta compara diferencias entre estados iniciales, estados finales, restricciones de camino y aplicabilidad de soluciones candidatas.

El segundo paso consiste en transformar la secuencia solución del problema viejo en una que satisfaga al nuevo. La transformación consiste en que a partir de un estado inicial (solución recordada en el paso 1), se llegue a un estado final (solución del actual), mediante la aplicación

[7] Este método trabaja de la siguiente manera: dado un conjunto de estados posibles del problema (incluido el estado inicial y el estado final), un conjunto de operadores (que transforma un estado en otro), una función diferencia (que mide la diferencia entre dos estados) y un conjunto de restricciones de camino, se mueve en el espacio de estados sucesivamente reduciendo la diferencia con el estado final. Cada solución es una secuencia de operadores y estados intermedios, incluyendo el estado inicial y el estado final, junto con las restricciones de camino que la solución satisface.

de los T-operadores; estos mapean una secuencia solución entera, en otra secuencia solución potencial. Este proceso, mediante la sucesiva aplicación de estos T-operadores, transita a través del espacio de soluciones acercándose a la secuencia solución del nuevo problema. Este acercamiento está medido por la "métrica de similitud".

En este T-espacio no hay restricciones de camino porque las soluciones intermedias no corresponden a problemas del mundo externo.

Este método se comporta de la siguiente manera teniendo en cuenta los 4 criterios anteriores:

. Dos problemas comparten aspectos significativos si son semejantes teniendo en cuenta la métrica de similitud.

. El conocimiento transferido a la nueva situación es la secuencia de acciones de la solución recuperada.

. El conocimiento es transferido copiando la solución recuperada y modificándola incrementalmente hasta satisfacer los requerimientos del nuevo problema.

. La selección de problemas pasados relevantes está restringida por el esquema de indexación de memoria y el proceso que confronta patrones parciales.

Si un tipo de problema ocurre con mucha frecuencia, es útil formar un plan generalizado en vez de solucionarlo cada vez por analogía.

Para formar tal plan, se aplica aprendizaje a partir de ejemplos, donde el concepto adquirido es un procedimiento de soluciones generalizadas y los ejemplos positivos serán aquellas soluciones generadas por el sistema que resuelve problemas analógicos, que funcionaron bien en el mundo externo; las que no funcionaron bien son ejemplos negativos (por la forma en que son generados estos ejemplos, ellos serán ejemplos negativos near-miss). Los ejemplos usados para formar un plan generalizado, son soluciones derivadas desde un mismo problema viejo en el proceso de transformación analógica.

El criterio de similitud puede ser modificado aplicando un proceso de refinamiento de parámetros, donde los casos en los que el proceso de transformación analógico pudo formar una solución para el nuevo problema sirven como ejemplos positivos para reforzar la métrica de similitud y los casos en que él falla sirven como ejemplos negativos.

Existen problemas que no pueden ser resueltos por analogía transformacional (aquellos en los que los aspectos significativos que comparten con las experiencias pasadas, no están en la solución, sino en los pasos intermedios de razonamiento). Estos pueden ser resueltos por una técnica analógica más sofisticada llamado **analogía derivacional** [1]. Esta usa más conocimiento operacional que la anterior, ya que se utiliza la información intermedia que se

genera cuando se formulan planes y se solucionan problemas. Esta información intermedia contiene el proceso de razonamiento usado para derivar la solución.

Un esquema de este método es:

1.- Cuando se soluciona un problema, se debe almacenar cada paso del proceso de solución (estructura de subobjetivos del problema, decisiones tomadas, punteros al conocimiento útil para la construcción de esta solución y la solución resultante).

2.- Si el proceso de razonamiento del problema dado (dado por las primeras fases de análisis) es comparable al de situaciones pasadas, recuperar todas las "pistas de razonamiento" (llamadas derivaciones) y aplicar el proceso de analogía derivacional. Si no fuera comparable, considerar la posibilidad de solucionarlo por analogía transformacional y si ella falla, intentar con razonamiento que no utiliza analogía.

3.- Se trata de aplicar la derivación recuperada a la situación actual. Para cada paso de la derivación, chequear si las razones para realizar ese paso todavía son válidas. Esto es hecho trazando dependencias entre la derivación recuperada y partes de la descripción del problema viejo o de suposiciones externas hechas cuando se solucionó el problema.

Si esas partes son también verdaderas en la nueva situación chequear el siguiente paso de la derivación. Si no fueran verdaderas, encontrar una nueva justificación para realizar ese paso y sino tomar otro camino (evaluar las alternativas no elegidas cuando se solucionó el problema viejo o abandonar analogía derivacional).

4.- Almacenar la diferencia de la derivación del nuevo problema con la derivación recuperada, para un posterior uso.

Este proceso es ilustrado en la figura 2.

La noción clave es reconstruir los aspectos relevantes de las situaciones en las que se solucionaron problemas pasados y de ese modo transferir conocimiento al nuevo escenario, donde ese conocimiento consiste de secuencias de decisiones y sus correspondientes justificaciones.

Dado los 4 criterios de comparación vemos que:

. Los problemas comparten aspectos significativos si sus análisis iniciales producen los mismos pasos de razonamiento (consideran los mismos aspectos y se toman las mismas decisiones).

. El conocimiento transferido a la nueva situación es la derivación de una situación pasada.

. El conocimiento es transferido reconsiderando las viejas decisiones: preservar las que son válidas en la nueva situación y reemplazar o modificar las que no lo son.

. El problema y sus derivaciones son almacenados en una memoria de episodios y la recuperación ocurre comparando las decisiones iniciales.

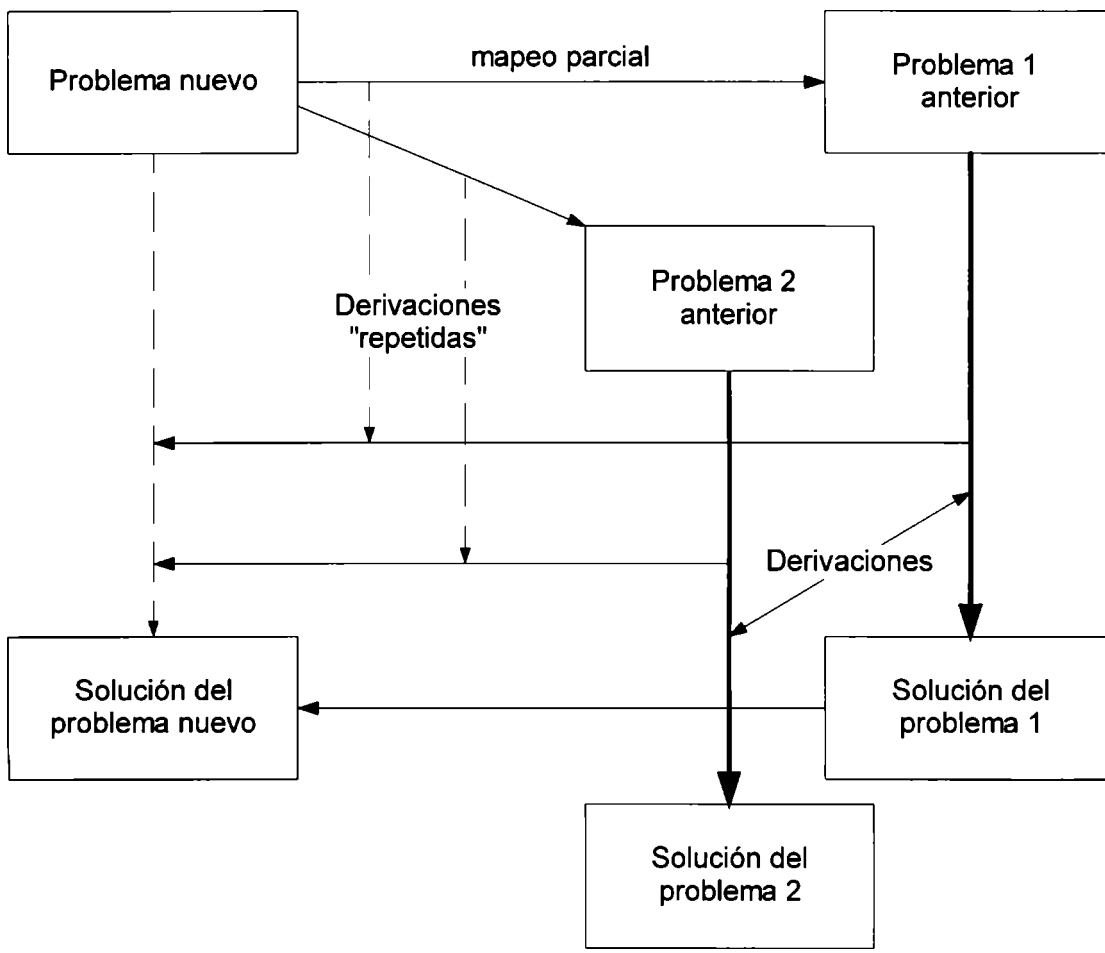


Figura 2: El proceso de analogía derivacional.

El mismo método inductivo general utilizado para inducir planes generalizados puede ser aplicado a diferentes partes de las pistas derivacionales: puntos de decisión, descomposición de problemas, selección de planes y métodos para evitar las causas de falla.

Otra técnica de aprendizaje en analogía derivacional es la formulación de reglas generales aplicables a un amplio rango de situaciones a partir de las secuencias derivacionales de problemas específicos. Para eso, primero se debe buscar en las pistas derivacionales, subsecuencias candidatas de acciones ocurridas con gran frecuencia. Luego, hay que encontrar una justificación para esas subsecuencias de acciones y a partir de esto se forma una regla general.

3.3) Métodos implementados

A continuación se describen algunos de los métodos correspondiente a este tipo de aprendizaje.

3.3.1) CARL

Según [3],CARL es un sistema que aprende acerca de la semántica de las sentencias de asignación para el lenguaje de programación BASIC. CARL combina aprendizaje por analogía y aprendizaje a partir de instrucción: es un sistema interactivo, donde las analogías, del tipo de las presentadas en los libros introductorios de programación de computadoras, son dadas por un instructor.

Cuando a CARL se le presenta un evento, primero busca en memoria una estructura causal (en el dominio del evento) que lo explique. Sino, usando una sentencia análogaa dada, él desarrolla una en el dominio objetivo: primero transforma la descripción del evento en el dominio objetivo en una descripción en el dominio base, por ejemplo:

dada la sentencia análoga:

"una variable es como una caja"

y el evento:

"¿cuál es el resultado de asociar un número a una variable?"

CARL transforma el evento al dominio base, resultando:

"¿cuál es el resultado de almacenar un objeto en una caja?"

Luego recupera de la memoria una estructura causal del evento en dicho dominio, por ejemplo:

```
modelo
   PONER-EN-CAJA
   roles-variables
      ROL-ACTOR (HUMANO)
      ROL-CAJA (CAJA)
      ROL-CONTENIDO (OBJETO)
```

acciones
(PTRANS
actor ROL-ACTOR
objeto ROL-OBJETO
desde (desconocido)
a (DENTRO de ROL-CAJA))
precondiciones
(no (DENTRO de ROL-CAJA está ROL-OBJETO))
(ROL-CAJA MAS-PEQUEÑO que ROL-OBJETO)
resultados
ROL-OBJETO DENTRO ROL-CAJA

y la adapta usando un proceso de mapeo top-down:

modelo
PONER-EN-VARIABLE
roles-variables
ROL-ACTOR (COMPUTADORA)
ROL-CAJA (VARIABLE)
ROL-CONTENIDO (NÚMERO)
acciones
(PTRANS
actor ROL-ACTOR
objeto ROL-OBJETO
desde (desconocido)
a (DENTRO-VAR de ROL-CAJA))
precondiciones
(no (DENTRO-VAR de ROL-CAJA hay ROL-OBJETO))
-eliminada
resultados
ROL-OBJETO DENTRO-VAR ROL-CAJA

El proceso de mapeo top-down debe determinar que relaciones del dominio base deben ser mapeadas al dominio objetivo. Las estructuras causales son la principal heurística utilizada por CARL para restringir el conjunto de posibles relaciones mapeadas por una analogía. Cuando una estructura causalmente conectada es encontrada en memoria para soportar una situación descripta del dominio base, él sólo considera para mapear, aquellas relaciones que participan

en esa estructura. Para esto, se requiere que las estructuras causales que soportan las situaciones del dominio base, estén disponibles al sistema de razonamiento análogoo.

Otra heurística para restringir el mapeo, utilizada por CARL, consiste en que los atributos de comparación no sean mapeados si los objetos en el dominio objetivo no pueden ser comparados bajo la misma escala relacional. No existe una escala para comparar un número con un variable, por lo tanto en el ejemplo se elimina la condición:

(ROL-CAJA MAS-PEQUEÑO que ROL-OBJETO)

El proceso de mapeo trata de formar estructuras en el dominio objetivo bajo las siguientes restricciones generales:

. Los predicados correspondientes deben ser de la misma clase (acciones, relaciones, etc.).
. Los predicados correspondientes en la estructura objetivo están relacionados por links causales o temporales, correspondientes a los links en la estructura del dominio base.
. Los slots correspondientes de los predicados analógicamente relacionados deben ser llenados consistentemente por "roles" correspondientes en las 2 estructuras relacionadas.

CARL mantiene un registro llamado AMAP que detalla las correspondencias de objetos, roles y predicados. Cuando son presentados nuevos ejemplos para los cuales no hay explicación en el dominio objetivo se puede usar la analogía anteriormente encontrada. Cada uno de estos subsiguientes usos también involucra mapear una descripción causal desde el dominio base al dominio objetivo. Para esto, la información en AMAP, del mapeo anterior, es usada en un proceso de mapeo inverso para construir una descripción del nuevo problema en el dominio base. Esta descripción es utilizada para recuperar una estructura causal en el dominio base que la explique. AMAP es luego utilizado para mapear esta estructura al dominio objetivo.

Las estructuras formadas por el proceso de mapeo pueden contener errores. Por lo tanto, estas deben ser depuradas de acuerdo a las sugerencias de correcciones dadas por el instructor.

A menudo es necesario mapear relaciones no idénticas: cuando se intenta mapear una relación tal que la relación en el dominio objetivo no comparte algunas de las propiedades de la relación en el dominio base, CARL forma una "relación virtual" en el dominio objetivo. Por

ejemplo, CARL formó la relación virtual DENTRO-VAR para especificar el almacenamiento de números en variables.

Existen problemas que no se pueden resolver con un solo consejo análogoo, por lo tanto puede ser necesario utilizar varios. Tener estructuras derivadas de analogías alternativas puede ser usado para reemplazar inferencias erróneas desarrolladas usando otra analogía o para modelar situaciones que no tienen equivalencia directa en otra analogía. Esto resulta en un modelo "mixto", donde cada analogía presta una función diferente. La capacidad de usar múltiples analogías reduce la cantidad de explicaciones y correcciones que deben ser dadas por el instructor.

3.3.2) NLAG

NLAG [4] es un programa de aprendizaje que incorpora el proceso de "inferencias útiles por analogía". Este proceso consta de 3 entradas: un problema, una teoría consistente, incapaz de resolver ese problema, y un consejo analógico. La idea es ampliar la teoría de tal manera de poder resolver el problema dado, utilizando el consejo analógico: encontrar hechos acerca de una fuente análoga, que nos permitan conjeturar los hechos del problema planteado, necesarios para resolverlo.

Una inferencia analógica útil es definida como:

$$\text{Th},\ A \sim B \vdash_{PT} Q(A)$$

donde no conocido: $\text{Th} \not\models Q(A)$
consistente : $\text{Th} \not\models \neg Q(A)$
común : $\text{Th} \models Q(B)$
útil : $\text{Th} \cup (Q(A)) \models PT$

Inferencia útil por analogía

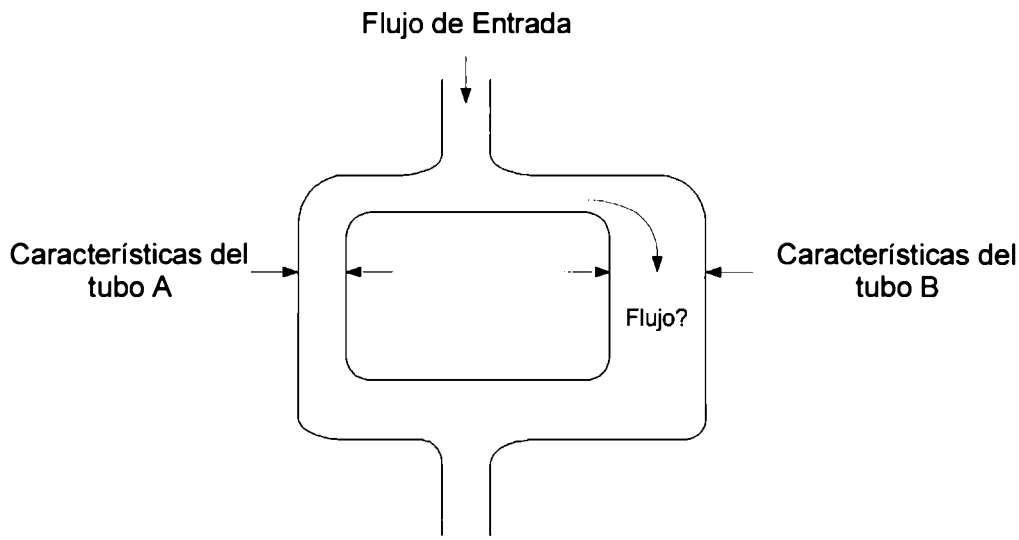
Donde:

A~B	: significa que B es el consejo análogoo para A.
No conocido	: significa que no son deductibles de la teoría.
Consistente	: se pide que la negación de las conjeturas no se pueda deducir a partir de la teoría, para evitar que cuando se agreguen a ella, produzcan inconsistencias.
Común	: pedimos que de la teoría se deduzcan los hechos, de la fuente análoga.

Útil : serán útiles las analogías si, unidas a la teoría, nos permiten resolver el problema planteado.

Los pasos de NLAG son: dado A~B, se debe encontrar una fórmula análogaa apropiada, $Q(x_1,...,x_n)$, tal que la fuente análoga la satisfaga, es decir que Th $|= Q(b_1,...,B,...,b_n)$ para algún conjunto de términos $\{b_i\}$. Luego se debe volver a instanciar esa fórmula usando el problema objetivo. El primer paso para esto es reemplazar B por A en Q, y se deberá encontrar los otros términos para llenar las otras n-1 posiciones, es decir encontrar el conjunto de términos $\{a_i\}$ tal que $Q(a_1,...,A,...,a_n)$ sea la conjetura necesaria para resolver el problema objetivo. Por ejemplo, dado el siguiente problema:

"Dada la siguiente conexión de tubos, sus características y el flujo de agua total de entrada, se desea saber el flujo a través de uno de los tubos"



y dado el consejo análogoo:

"El flujo de agua es como la corriente"

se debe encontrar $Q(x_1,...,x_n)$ tal que: Th $|= Q(b_1,...,Electricidad,...,b_n)$. Se debe encontrar los términos $\{a_i\}$ tal que $Q(a_1,...,Hidráulica,...,a_n)$ sea la conjetura necesaria para resolver el problema objetivo.

La inferencia análogaa lleva a considerar diferentes conjuntos de conjeturas. El objetivo es orientar la búsqueda para encontrar la "mejor".

La principal heurística para podar el espacio de posibles analogías se basa en la idea de "abstracciones"; fórmulas reusables generales, que representan relaciones abstractas que codifican soluciones a problemas previos. Se consideran reusables porque ellas han sido útiles

en diferentes situaciones. Una abstracción surge de hacer un apropiado reemplazo de constantes por variables en un conjunto de hechos relevantes para un problema dado.

El sistema NLAG usa abstracciones predefinidas, él no genera nuevas.

Cuando NLAG busca una fórmula análogaa, sólo considerará fórmulas que hayan sido declaradas como una abstracción. Esta heurística sugiere que una buena analogía útil es la que conjetura los hechos necesarios para instanciar una abstracción.

Como se vio antes, se debe encontrar el conjunto de términos $\{a_i\}$, esto es equivalente a instanciar una abstracción de manera adecuada para solucionar el problema dado.

El uso de abstracciones todavía deja un número grande de analogías a considerar. Existen otras heurísticas para decidir cuales considerar, en que orden y para encontrar la apropiada instanciación para cada una:

Heurística justificación Kernel (H_{jk}): sirve para determinar cuales abstracciones considerar y en que orden. H_{jk} sugiere un problema relacionado al problema objetivo en el dominio del análogo fuente. Bajo la hipótesis que una determinada abstracción que es útil para solucionar el problema fuente, también lo será para solucionar el problema objetivo; nuestro objetivo es encontrar la abstracción que tenga más en común con el problema fuente. H_{jk} busca la abstracción S, la cual maximiza:

$$\alpha \cap \beta$$

donde α es el conjunto de hechos necesarios para resolver el problema análogo fuente y β es el conjunto de hechos necesarios para instanciar a S en el dominio del problema fuente.

Heurística dominio común (H_{cd}): sirve para podar el espacio de posibles instanciaciones de una abstracción dada. Considera las instanciaciones donde los términos $\{a_i\}$ pertenecen al mismo dominio.

Estas 2 heurísticas son llamadas reglas basadas en abstracciones.

Las 3 heurísticas siguientes están relacionadas al concepto de "proximidad", o sea preferir las abstracciones que están "más cerca" del mundo conocido. Una analogía estará más cerca cuando la cantidad de restricciones adicionales (nuevas conjeturas) impuestas sobre el mundo conocido (la teoría) es menor. Estas son llamadas reglas basadas en restricciones minimales:

Heurística menor cantidad de conjeturas (H_{fc}): sirve para ordenar las instanciaciones de una abstracción dada. Prefiere las instanciaciones que agregan la menor cantidad de conjeturas posibles.

Heurística abstracciones más generales (H_{MGA}): ordenan el espacio de posibles abstracciones. Dada las abstracciones S_1 y S_2, H_{MGA} es definida como:

Si S_1 => S_2

Entonces preferir S_2 sobre S_1

prefiere a S_2 porque es más general que S_1, esto significa que S_2 debe realizar menos conjeturas que S_1.

Heurística términos "aprendibles" (H_{FT}): poda el espacio de instanciaciones de abstracciones. Esta heurística permite evitar casos incomunes. La teoría debe proveer alguna razón para sugerir un término en una instanciación. NLAG sólo considera términos que tengan "soporte": cada término debe cumplir algunas propiedades. Si algún término no cumple al menos una de ellas, NLAG lo ignora. Por ejemplo, que un término deba cumplir la propiedad de ser una función de un argumento.

Cuando haya conflicto entre 2 heurísticas, se prefiere siempre una regla basada en abstracciones sobre una regla basada en restricciones minimales.

Continuando el ejemplo, las heurísticas anteriores consideraron la siguiente abstracción como la más adecuada para solucionar el problema:

RKK(t, c, r, l) =

K1(t) "La suma algebraica de todos los t (corriente) a través de una estructura del tipo **Y** es 0"

K2(c) "La suma de todos los c (voltaje perdido) en un circuito cerrado es 0"

Ohms(t, c, r, l) "c (voltaje perdido) = t (corriente) * r (resistencia del conductor)

Conservación de verdad(t, l) "t (corriente) es conservada a través de l (conductor)"

y a la siguiente instanciación:

Q_{RKK}(FlujoDeAgua, PresiónPerdida, CaracterísticasDelTubo, Tubo)

como analogía a la instaciación el dominio fuente:

Q_{RKK}(Corriente, VoltagePerdido, ResistenciaDelConductor, Conductor)

3.3.3) Clint-Cia [2]

Este sistema (descripo en [2]), está formado por un subsistema interactivo que aprende conceptos, Clint, junto con un sistema asistente llamado Cia. Cint-Cia es un "learning apprentices": una ayuda interactiva para construir y refinar una base de conocimientos durante la operación normal del sistema.

Clint es un sistema incremental que aprende a partir de ejemplos, usa conocimiento, genera ejemplos, cambia su sesgo (ver definición pág 7), se recupera desde estados de error e identifica conceptos en el límite.

Cia es un asistente de Clint, que aplica técnicas de aprendizaje por analogía. Interactivamente, propone nuevos conceptos (inducción constructiva) de interés del usuario, a ser puestos en la base de conocimiento y en el lenguaje de descripción de conceptos.

Los 2 principales requerimientos para un sistema que aprende conceptos en el contexto de "learning apprentices" son: que sea amigable al usuario y que sea oportuno, es decir que tome la iniciativa de aprender cuando le sea posible.

Existen 2 tipos de oportunidades:

* Oportunidades manejando ejemplos: ocurre cuando el usuario ingresa un ejemplo positivo no cubierto o un ejemplo negativo cubierto. Ellas son manejadas por Clint.
* Oportunidades manejando conceptos: ocurre cuando el sistema es capaz de proponer al usuario nuevas definiciones de predicados interesantes. Ellas son manejadas por Cia.

El sistema usa un framework lógico: los conceptos son predicados, una definición de concepto es un conjunto de cláusulas de Horn que definen un predicado y los ejemplos son instancias básicas (Ground) de esos predicados.

Clint usa una serie ordenada de definiciones de lenguajes $L_1,\ldots,L_n,\ldots$ tal que todo predicado expresable en L_i es expresable también en L_{i+1}. Estos lenguajes son calculados dinámicamente cuando es necesario.

Clint trata los tipos de oportunidades manejando ejemplos (ejemplos positivos no cubiertos y ejemplos negativos cubiertos), de diferente manera:

a) Positivo no cubierto: cuando el sistema recibe un ejemplo positivo no cubierto, él construye una nueva cláusula que cubra el ejemplo y la agrega a la base de conocimiento.

Para hacer esto, el sistema construye una cláusula de partida en L_i, cláusula máximamente específica que cubre al ejemplo. Si esta cláusula de partida cubre algún ejemplo negativo, como Clint sólo considera generalizaciones, esta debe ser descartada. Selecciona una nueva cláusula de partida en el primer lenguaje (siguiendo la serie $L_1,...,L_n,...$) que no cubra ejemplos negativos. Para obtener el próximo lenguaje en la serie, existen 2 operaciones: permitir más variables cuantificadas existencialmente y permitir relaciones entre las variables previamente introducidas. El primer lenguaje, L_0, sólo permite cláusulas tal que toda variable que ocurre en el cuerpo también ocurre en la cabeza. Por ejemplo, dada una base de conocimiento con hechos acerca de los predicados "masculino", "femenino", "padre" y "ancestro" y el siguiente ejemplo positivo no cubierto:

abuela(Alicia, Pedro)

Clint construye en L_0 la siguiente cláusula de partida:

abuela(G,C) ← masculino(C) & femenino(G)

Si ella cubre un ejemplo negativo, se construye una cláusula de partida en L_1:

abuela(G,C) ← masculino(C) & femenino(G) & padre(F,C) &
ancestro(M,C) & ancestro(G,H)

En la cláusula de partida seleccionada pueden existir relaciones irrelevantes, estas pueden ser reconocidas preguntándole al usuario.

Una vez que una cláusula de partida consistente ha sido construida (si Cia no encontró una descripción correcta del concepto), esta es generalizada usando una búsqueda de lo específico a lo general, eliminando condiciones y haciendo **preguntas miembro** al usuario. Una pregunta miembro pide la clasificación de un ejemplo para un concepto. Este ejemplo es formado de manera tal que no sea cubierto por la cláusula de partida y sea cubierto por la generalización. Si el usuario responde que el ejemplo es positivo, la generalización es válida; sino, se debe tomar otra generalización. Por ejemplo, dada la cláusula de partida anterior, una posible generalización de ella es:

abuela(G,C) ← femenino(G) & padre(F,C) & ancestro(M,C) & ancestro(G,H)

una pregunta miembro para dicha generalización es:

¿abuela(Susana,Sofia) es verdadero?

si la respuesta del usuario es afirmativa, esta generalización es válida y se continúa generalizando a partir de ella.

b) Ejemplo negativo cubierto: cuando existe un ejemplo negativo cubierto, significa que existe una cláusula incorrecta en la base de conocimiento. Para localizarla, el sistema construye el árbol de demostración para el ejemplo negativo y lo analiza, haciendo preguntas al usuario si es necesario. Cuando la localiza, es eliminada de la base de conocimiento.

El sistema, para mantener la consistencia, debe verificar si los ejemplos positivos que eran cubiertos por la cláusula eliminada, siguen estando cubiertos. También deben ser verificados todos los predicados los cuales fueron aprendidos involucrando la cláusula borrada.

Cia en cada oportunidad manejando conceptos, propone al usuario varias definiciones de predicados, quien tiene que darle nombre a los predicados si son relevantes para el dominio y descartarlos en caso contrario. La técnica Cia emplea las similitudes observadas, derivando para cada descripción de concepto aprendida, un esquema de segundo orden. Un esquema de segundo orden es una cláusula de Horn abstracta, donde los nombres de predicados son interpretados como "variables_predicados" cuantificados existencialmente.

La principal suposición hecha por Cia es: las descripciones de conceptos son a menudo similares en el sentido que ellas son instancias del mismo esquema de segundo orden.

Clint llama a Cia en 2 diferentes momentos:

* Cuando Clint ha encontrado una cláusula de partida consistente cubriendo un cierto ejemplo positivo. En ese caso Cia intenta aprender una cláusula correcta para el concepto, haciendo **preguntas de cláusulas**. Una pregunta de cláusula interroga al usuario para ver si una cláusula es correcta para dicho concepto. Esta cláusula es formada instanciando un esquema de segundo orden que confronte[8] con la cláusula de partida consistente. Por ejemplo, dada la cláusula de partida:

$$padre(F,C) \leftarrow masculino(F) \,\&\, masculino(C) \,\&\, ancestro(F,C)$$

y el esquema de segundo orden (que confronta con ella):

$$\exists\, p, q, r \,/\, p(X,Y) \leftarrow q(X) \,\&\, r(X,Y)$$

la pregunta de cláusula resultante que se le hace al usuario es:

¿padre(X,Y) ← masculino(X) & ancestro(X,Y) es la definición del concepto que se quiere aprender?

Si el usuario responde que dicha cláusula es correcta para el concepto que se está aprendiendo, esta será guardada en la base de conocimiento. Sino, esta cláusula puede

[8] Un esquema de segundo orden S confronta con una claúsula c si: $\exists$ una sustitución θ tal que cabeza(S) = cabeza(c) y cuerpo(S)$\theta \subset$ cuerpo(c).

definir otro concepto. Para ver si es así, se hacen **preguntas de nuevo término** al usuario. Una pregunta de nuevo término interroga al usuario sobre un nombre de predicado para una definición dada de él. Si el concepto objetivo no es encontrado (por Cia) Clint continuará intentando aprender el concepto.

* Después que Clint aprendió una cláusula para un concepto, Cia intentará inventar o descubrir nuevos conceptos haciendo preguntas de nuevo término. Estas definiciones serán formadas instanciando los esquemas de segundo orden que sólo confronten parcialmente[9] con la cláusula de partida del predicado aprendido. Por ejemplo, dada la cláusula de partida:

padre(F,C) ← masculino(F) & masculino(C) & ancestro(F,C)

y el esquema de segundo orden (que confronta parcialmente con ella):

∃ p, q, r / p(Y,X) ← q (X) & r (X,Y)

la pregunta miembro resultante que se le hace al usuario es:

¿p(Y,X) ← masculino(X) & ancestro(X,Y) es la definición de un nuevo concepto?

¿Cuál?

Una respuesta posible es:

p = "hijo"

Las preguntas que realiza Clint-Cia al usuario pueden ser consideradas como una forma de implementar aprendizaje a partir de instrucción.

Algunos discuten que Cia solo realiza una forma muy débil de analogía (Kodratoff, Morik, etc.), porque la similitud entre los problemas sólo es sintáctica. Considerando que la analogía consiste en solucionar un problema a partir de las soluciones de problemas anteriores similares, podemos pensar que Cia realiza analogía ya que encuentra soluciones instanciando esquemas de segundo orden, los cuales fueron construidos a partir de soluciones de problemas anteriores.

---

9 Un esquema de segundo orden S confronta parcialmente con una claúsula c si: ∃ una sustitución θ tal que cuerpo(S)θ ⊂ cuerpo(c).

3.4) Comparación y conclusión de métodos por analógicos

A partir del análisis de los métodos descriptos anteriormente, creímos útil mostrar como cada uno de los métodos trata las características relevantes de aprendizaje por analogía. La idea del cuadro es guiar al usuario de este survey a elegir el método que mejor se adecue a sus necesidades.

* Podemos ver que las abstracciones de NLAG y las estructuras causales de CARL pueden ser comparables. Cuando se da un problema objetivo y un problema análogo fuente, CARL encuentra una estructura causal que soporte el problema análogo fuente y mapea las relaciones intervinientes en ella para producir una nueva estructura causal para el problema objetivo, mientras NLAG encuentra una instanciación de una abstracción que soporta el problema fuente, y conjetura los hechos necesarios para instanciar la misma abstracción de manera tal de soportar al problema objetivo. Podemos ver que CARL crea nuevas estructuras causales a medida que soluciona nuevos problemas, mientras NLAG sólo permite instanciar abstracciones preexistentes.

  Por otro lado pueden ser comparadas las abstracciones de NLAG y los esquemas de segundo orden de CLint-Cia. Ellos pueden ser vistos como esqueletos que pueden ser instanciados para: resolver un problema en NLAG y generar nuevos conceptos en Clint-Cia.

* Existen sistemas a los cuales se les da el consejo análogo (situación semejante que sirve como base para resolver la situación objetivo), mientras otros sistemas buscan ese consejo en su propia experiencia. Estos últimos necesitan saber, de alguna forma, cuales son los problemas semejante útiles. Como ejemplos de los primeros sistemas estarían CARL y NLAG y de los segundos, Clint-Cia.

* NLAG es un proceso de naturaleza top-down, ya que cada abstracción es un modelo específico el cual sirve para enfocar la búsqueda. CARL también usa una forma de razonamiento análogo top-down, ya que las estructuras dan un conjunto de restricciones top-down sobre las relaciones consideradas a mapear. Clint-Cia usa un razonamiento análogo mixto, bottom-up y top-down, ya que para generar un concepto tiene en cuenta cláusulas de partida (bottom-up) y esquemas de segundo orden (top-down).

* Existen métodos que intentan aprender por procedimientos no análogos cuando no pueden hacerlo por analogía, mientras otros, lo hacen para obtener más rápidamente la solución. NLAG usa analogía con el propósito de conjeturar hechos no derivables de la teoría. CARL intenta aprender por analogías dadas por un instructor.

Clint-Cia intenta aprender un concepto por analogía primero, si no lo logra, aplicará aprendizaje a partir de ejemplos.

	Estructuras generales reusables	¿Como referenciar al problema análogo?	Consejo análogo	Estrategia de control	¿Aprende sólo por analogía?	Específico vs general
Clint-Cia	Reusa esquemas de segundo orden, instanciándolos	Sólo con esquemas de segundo orden, que se generen a partir de él	Lo busca en su experiencia	Mixto, bottom-up y top-down	No. Cuando no puede aprender por analogía, intenta a partir de ejemplos	General
NLAG	Rehusa abstracciones, instanciándolas	Busca una instanciación de una abstracción que lo soporta	Se le da	Top-down	Si	General
CARL	No existen estructuras generales reusables, genera nuevas estructuras causales	Busca la estructura causal que lo soporta	Se le da	Top-down	Si	Específico

3.5) Razonamiento basado en casos

El proceso de analogía derivacional puede ser propuesto como un medio para dotar a los sistemas expertos de la habilidad de razonar a partir de casos. Es sabido que la tarea de extraer desde los expertos humanos el conocimiento asociado con su habilidad para resolver determinados problemas no es para nada fácil. Por otro lado si los mismos expertos humanos producen reglas deductivas generales que expliquen su comportamiento, generalmente quedarán casos relevantes sin considerar. Por lo tanto, una aproximación más eficiente es requerirle al experto que reporte aspectos relevantes del dominio que tuvo en cuenta para solucionar un problema. Esto da las pistas derivacionales externas que una máquina de inferencia de analogía derivacional puede usar para solucionar efectivamente futuros problemas similares. Esta forma de aplicar analogía derivacional es llamada razonamiento basado en casos [26].

Un caso es una pieza de conocimiento conceptualizado (conocimiento específico ligado a un contexto) representando una experiencia que enseña una lección fundamental para alcanzar los objetivos del sistema.

Los casos son almacenados en una librería de casos para su futuro uso. En ella debe haber casos asociados a soluciones normales y anormales. Los casos normales (es decir, los que no tienen diferencia con lo que se espera que pase) son almacenados una única vez, para evitar que las librerías contengan conocimiento redundante.

La librería de casos está ordenada de acuerdo a los "índices" de los casos. Un índice es el contexto en el cual un caso puede enseñar su lección; o sea bajo que circunstancias es apropiado recuperarlo.

El aprendizaje en razonamiento basado en casos ocurre de dos formas: a través de la acumulación de nuevos casos y a través de la asignación de índices.

Los casos pueden servir para:

- Proveer sugerencias de soluciones de problemas: soluciones que serán adaptadas para encajar en una nueva situación.
- Proveer contexto para la interpretación de una situación: evidencia concreta a favor o en contra de alguna solución o interpretación de una situación.

Los procesos del razonamiento basado en casos son:

1.- Recuperar casos: este es el paso principal, en el cual se debe seleccionar uno o varios casos a partir de los cuales se va a razonar en los próximos procesos.

En este proceso se debe determinar el problema de indexación: recuperar casos aplicables en el momento apropiado. Este ha sido tratado como el problema de asignar "rótulos", llamados índices, a los casos; estos designan bajo que condiciones cada uno de los casos puede ser usado para sacar inferencias útiles.

2.- Proponer una solución: este paso, para solucionar problemas, involucra seleccionar soluciones de problemas viejos o partes de ellas, como una solución para el nuevo problema; y en interpretación, involucra particionar los casos recuperados de acuerdo a las interpretaciones que predicen y basándose en eso, asignar una interpretación inicial al problema.

3.- Adaptación (sólo para solucionar problemas): generalmente las viejas soluciones deben ser modificadas para que encajen con la nueva situación.

4.- Justificación y crítica (sólo en interpretación): En el proceso de justificación, se buscan similitudes que justifiquen los resultados deseados y diferencias que impliquen que otros factores deben ser tomados en cuenta.

Este paso es, a veces, seguido por un paso de crítica, en el que se testean los resultados obtenidos.

5.- Evaluación: en este paso, los resultados del razonamiento son probados en el mundo real (directamente aplicados a él o través de simulación).

6.- Modificación de memoria: un nuevo caso debe ser apropiadamente almacenado en la memoria de casos para su futuro uso. Un caso está formado por el problema, su solución y cualquier hecho subyacente o razonamiento secundario que el sistema sepa como usar.

En este paso, el proceso más importante es elegir la manera de indexar el nuevo caso en la memoria.

3.6) Métodos de razonamiento basado en casos

Es esta sección se describen algunos de los métodos implementados de razonamiento basado en casos.

3.6.1) CHEF

CHEF (desarrollado por Hammend [27]), es un planificador basado en casos en el dominio de la creación de recetas de cocina. El recibe como entrada una conjunción de objetivos y su salida es un plan. Por ejemplo, se desea crear una receta que cumpla los siguientes objetivos:

- Incluir carne y
- Incluir brócoli y
- Que sea una comida frita.

El primer paso es recordar una receta que satisfaga la mayor cantidad posible de objetivos; por eso los planes están indexados por los objetivos que satisfacen. Por ejemplo, la receta recuperada para el ejemplo anterior es "carne con arvejas", donde los objetivos son:

- Incluir carne y
- Que sea una comida frita y
- Incluir arvejas (las arvejas son vegetales como el brócoli).

El segundo paso es adaptar la receta recordada, realizando dos fases:

- Reinstanciar el plan viejo: sustituyendo los objetos viejos por los nuevos, basado en la similitud entre objetos. Por ejemplo, reemplazar arvejas por brócoli.
- Aplicar los "críticos de objetos" de propósito especial: es la forma de codificar conocimiento de procedimientos especiales. Cada crítico está asociado con un objeto del dominio e implica agregar al plan pasos de preparación especiales. Por ejemplo, un crítico de objeto del brócoli es:

  Antes del paso: freír los ingredientes.

  Hacer: cortar el brócoli en trozos.

Después de estos pasos, se tiene un plan completo, por ejemplo:

Problema:

- Incluir carne y
- Incluir brócoli y
- Que sea una comida frita.

Solución:

Ingredientes:

- Ingr1: brócoli.

Acciones:

- Act1: cortar objeto (Ingr1) en trozos.

pero para poder aprender, se necesita saber si el plan funciona bien o no. El plan formado es "corrido" a través de un simulador, que realiza los pasos del plan. El provee la información necesaria que permite comparar los resultados esperados con los resultados obtenidos de la simulación.

Cuando estos resultados no coinciden, CHEF utiliza una estructura llamada TOP, en la cual hay descripciones de situaciones generales de fallas de planes y las estrategias de reparación asociadas a cada una. Además, para cada estrategia, TOP tiene almacenados casos que ella solucionó. En el ejemplo anterior, un resultado esperado no satisfecho es:

- Que el brócoli esté crocante.

CHEF construye una explicación causal de la falla y ella es usada para encontrar en la estructura TOP las estrategias adecuadas para solucionarla. La explicación construida por CHEF para el ejemplo anterior es la siguiente:

"En el paso que se fríen los ingredientes, la carne suelta líquido y empapa al resto de los ingredientes"

Las estrategias encontradas en TOP para solucionar este tipo de falla son:

- Dividir los pasos del plan.
- Encontrar un plan alternativo.

Luego, comparando el caso actual con los casos almacenados en cada estrategia, se elige la más adecuada. CHEF elige la primera estrategia, y su aplicación resulta en que el paso:

- Freír la carne junto con el brócoli.

sea reemplazado por los pasos:

- Freír la carne.
- Freír el brócoli.
- Unir la carne frita y el brócoli frito.

Para evitar que la misma falla suceda en el futuro, CHEF agrega un paso adicional llamado anticipación de fallas. Para esto hay que encontrar los "predictores" de la falla, que actuarán como índices del plan fallido, y este será usado como caso de advertencia. Cuando se presente un nuevo problema con objetivos similares a los de un plan fallido, que cumpla los predictores de la falla, se agregará a la descripción del problema la sentencia "evitar dicho tipo de falla". De esta manera, CHEF aprende a evitar errores pasados.

3.6.2) CASEY

CASEY (desarrollado por Koton [27]), es un diagnosticador basado en casos: recibe como entrada una descripción de un paciente y su salida es una explicación causal de los desordenes que él tiene. Por ejemplo, una descripción de un paciente es:

Sexo: Masculino.

Náuseas: Ausente.

Pulsaciones: 90 por minuto.

El está construido sobre un programa que diagnostica fallas del corazón. Cuando se presenta un nuevo paciente, CASEY se fija si en su librería de casos tiene un caso similar para diagnosticar al paciente; si no es así, le pasa el problema al programa que diagnostica fallas del corazón. Este diagnostica el caso y le retorna el resultado a CASEY, para que pueda ser usado en el futuro.

CASEY usa dos pasos en el proceso de diagnóstico:

1.- Buscar en memoria casos similares y usar reglas de evidencia basadas en modelos para determinar cuales, de los que confrontan parcialmente, son suficientemente similares para sugerir un buen diagnóstico.

Las reglas de evidencia especifican bajo que circunstancias las diferencias pueden ser reconciliadas. Estas intentan confrontar los valores de las características correspondientes en los dos casos. Lo que permite determinar esta correspondencia, son las heurísticas del modelo causal que nos indican cuando dos características tienen el mismo rol funcional. Si las reglas de evidencia no permiten reconciliar las diferencias, el caso es descartado.

Por ejemplo, un caso similar al anterior es:

Sexo: Masculino.

Náuseas: Ausente.

Pulsaciones: 96 por minuto.

donde la diferencia en la característica pulsaciones es reconciliada con la siguiente regla de evidencia:

"Igual región cualitativa"

esta regla concilia la diferencia entre los valores de una característica si ellos pertenecen a la misma región cualitativa (según el modelo causal).

2.- Aplicar reglas de reparación basadas en modelos para adaptar el diagnóstico, sugerido por el caso viejo, a la nueva situación. Existe una asociación entre una regla de reparación y una o más reglas de evidencia. CASEY mantiene pistas de las reglas de evidencia que aplicó en el paso anterior y luego, a partir de eso, aplica las reglas de reparación correspondientes para adaptar la solución.

La regla de reparación asociada a la regla de evidencia aplicada en el ejemplo es:

"Sustituir evidencia"

ella reemplaza el valor anterior de la característica conciliada por el nuevo valor en la explicación causal (90 por 96 pulsaciones por minuto).

3.6.3) JULIA

JULIA (desarrollado por Hinrichs y Kolodner [27]), es un planificador basado en casos que trabaja en el dominio de planificación de comidas. Los problemas son descriptos de acuerdo a las restricciones que él debe cumplir y las soluciones describen una estructura que satisface la mayor cantidad de dichas restricciones.

JULIA diseña una comida combinando procesos de razonamiento basados en casos con procesos que fijan y propagan restricciones.

Las restricciones pueden ser derivadas del conocimiento general de comidas o derivadas del nuevo problema (impuestas por el usuario). Por lo tanto, las fuentes de conocimiento son, además de los casos, las especificaciones particulares del nuevo diseño (requerimientos del nuevo problema) y conocimiento general dado por los "prototipos de objetos". Un prototipo especifica la estructura de una comida y las relaciones entre sus partes (por ejemplo, un prototipo para comidas americanas).

Cuando se le presenta un nuevo problema a JULIA, primero intenta encontrar la información faltante de la especificación del problema a través de inferencias y preguntas al usuario. Luego determina el "framework" de la solución: primero encontrando un caso que confronte con la especificación del problema y si no existe, elige un prototipo de objeto apropiado. Por ejemplo, un problema presentado a JULIA puede ser:

- Comida vegetariana.
- Comida barata.
- Comida fácil de preparar.
- Incluir queso y tomate como ingredientes principales.

El uso de queso y tomate sugieren dos tipos de comidas, mexicana e italiana. Por lo tanto, una posible pregunta al usuario es:

¿Cuál de las dos prefiere?

Suponiendo que él responde que prefiere la comida italiana, un framework para solucionar este problema puede ser el prototipo de una comida italiana, que incluye:

- Un plato de antipasto.
- Un plato de pasta.
- Un plato principal.
- Un postre.

Luego los métodos que tratan las restricciones las propagan a través del framework, para reconciliar las diferencias entre él y los requerimientos de la nueva situación. Esto implica que en algunas partes del framework pueda haber conflictos; por lo tanto, estas deben ser adaptadas para que satisfagan las restricciones. Además otras partes deben ser completadas. JULIA llena las partes incompletas, usando razonamiento basado en casos; o sea, recuerda la parte correspondiente de un caso que satisfaga la mayor cantidad posible de las restricciones impuestas sobre esa parte de la solución y luego la adapta apropiadamente, pidiendo confirmaciones al usuario. En el ejemplo anterior, un caso similar le sugiere a JULIA lazaña como plato principal.

JULIA realiza dos tipos de adaptaciones, basadas en la violación de restricciones:

- Sustitución: implica que un componente de un ítem sea reemplazado por otro. Para esto se necesitan heurísticas que usan conocimiento sobre la importancia relativa de los diferentes componentes de un ítem. Por ejemplo, como un ingrediente de la lazaña es la carne y una restricción es que la comida debe ser vegetariana, se sustituye este plato por una "lazaña vegetariana".
- Cambios en la estructura: implica agregar o eliminar algún ítem del framework. Para esto se necesitan heurísticas de adaptación de propósito especial. Como se tiene una heurística que determina que no se pueden repetir los ingredientes en los distintos platos de una comida, del framework del ejemplo anterior se elimina el plato de pasta porque la lazaña es una pasta.

Como en general no se puede encontrar un caso a partir del cual se pueda construir toda la solución, JULIA divide el problemas en subproblemas, y resuelve cada uno usando razonamiento basado en casos.

El proceso normal que JULIA realiza para construir la solución puede ser alterado porque:

- El usuario da una nueva restricción: en este caso JULIA debe reparar la solución parcial si esta no la satisface.
- Se recupera un caso que advierte del peligro que ocurra una falla: en este caso JULIA debe considerar si esta falla puede ocurrir en la situación actual, y si así fuera impone una nueva restricción para evitarla.

3.6.4) HYPO

HYPO (desarrollado por Ashley y Rissland [27]), es un interpretador basado en casos que trabaja en el dominio de la ley. Toma como entrada una situación legal y crea como salida un argumento para su usuario. Puede tomar la posición del demandante o del demandado.

Los pasos en el razonamiento de HYPO son:

1.- Analizar la nueva situación para encontrar factores relevantes: HYPO usa "dimensiones" para representar las características predictivas de los casos, que son derivadas de las características visibles. Las dimensiones mantienen cuatro tipos de conocimiento:

- Prerequisitos: determinan si una situación puede ser descripta por la dimensión.
- "Slots" focales: indican los descriptores que son importantes para determinar la magnitud de la dimensión.
- Métodos de comparación: especifica como usar la información del slot focal, para la comparación.
- Índices: las dimensiones indexan los casos y señalan los casos que ellas indexan.

Un factor relevante es una dimensión que influencia el resultado del caso. La nueva situación es analizada testeando los prerequisitos de las dimensiones para ver cuales la describen.

El ejemplo que se va a desarrollar es un ejemplo sencillo de un "pleito" de la vida cotidiana:

> "Jorge está próximo a cumplir 13 años, quiere ir al cine a ver una película calificada para mayores de 13 años y sus padres no lo dejan"

Los factores relevantes de este pleito son:

. Edad

. Madurez.

. Calificación de la película.

2.- Recuperar casos que comparten esos factores: se recuperan los casos indexados por las dimensiones encontradas en el paso anterior. Los casos recuperados para el ejemplo anterior son:

Caso 1: "A la hermana mayor de Jorge la dejaron ir a ver la misma película."

Caso 2: "A Pedro que tiene menos de 13 años no lo dejaron ir a ver otra película calificada para mayores de 13 años"

Caso 3: "A Manuel que tiene la misma edad que Jorge lo dejaron ir a ver la misma película"

3.- Ubicar los casos recuperados respecto de la nueva situación: para esto se usa una "red de pleitos", que es un grafo que organiza los casos recuperados en el paso 2; de modo que sus similitudes y sus diferencias estén explícitas. En este paso se dividen los casos recuperados en los que soportan la posición del sistema y los que soportan la posición opuesta. En el ejemplo, los casos 1 y 3 soportan la posición de Jorge y el caso 2 soporta la posición opuesta.

4.- Seleccionar los casos más significativos para cada conjunto: los casos más significativos serán aquellos que compartan más factores relevantes (dimensiones) con la nueva situación. La red de pleitos ayuda para este propósito.

5.- Generar el argumento: un argumento tiene tres componentes:

a) Una "afirmación" es hecha en favor de la posición en la que está el sistema, apoyando un resultado. Esta formada por tres partes:

- Una conclusión apoyando el resultado.
- Una justificación de la conclusión a través de la citación de un caso previo, el más significativo.
- Una razón fundamental de la justificación a través de las similitudes entre la nueva situación y el caso citado.

La afirmación en el ejemplo es:

Conclusión: a Jorge lo tiene que dejar ir a ver al película.

Justificación: a su hermana la dejaron.

Razón fundamental: la calificación de la película.

b) Una respuesta en favor del oponente: debe tener alguno de los siguientes componentes:

- Una sentencia que distinga la nueva situación del caso citado como justificación. Esta puede estar dada porque:

i) Algunas dimensiones compartidas, entre el caso nuevo y el caso dado como justificación, tengan valores distintos, o

ii) Existan dimensiones no compartidas, entre el caso nuevo y el caso dado como justificación.

- Una citación de un contraejemplo del caso citado como justificación: son casos que comparten alguna dimensión con la nueva situación y la justificación, pero que tiene un resultado opuesto.

La respuesta del oponente en el ejemplo es:

"La justificación no es válida porque la hermana de Jorge es mayor y más madura."

c) Una refutación en favor de la posición en la que está el sistema, impugnando la respuesta. Esta puede ser dada por: sentencias distintivas o contraejemplos de la respuesta (por ejemplo, si la respuesta indica diferencias sobre una dimensión, esta puede ser refutada encontrando un caso con la misma diferencia que soporte la conclusión de la afirmación original). Una refutación para el ejemplo es el caso 3.

6.- HYPO crea casos hipotéticos, a partir de casos reales, para testear el análisis hecho hasta ese momento. Estos son particularmente útiles para testear el argumento.

3.6.5) PROTOS

PROTOS (desarrollado por Porter, Bareiss, Holte y Weir [27]) implementa clasificación basada en casos y adquisición de conocimiento basada en casos, en el dominio de la audiología.

Dada una descripción de los síntomas y los resultados de los test de algún paciente, PROTOS determina su desorden audiológico.

PROTOS utiliza cuatro clases de "links":

- Links recordativos: asocia características con categorías.
- Links de prototipos: conecta los casos más típicos de una categoría con ellas.
- Links diferencias: registras diferencias importantes entre los casos.
- Links censores: excluye conexiones dadas por otros links.

Los pasos de la clasificación basada en casos son:

1.- Hacer una hipótesis inicial: viendo como confrontan las características del nuevo caso con las características de cada categoría, usando los links recordativos, se elige un conjunto de categorías probables y entre ellas, se toma la mejor como hipótesis.

2.- Seleccionar el caso más representativo de la categoría elegida: los links de prototipos están ordenados de acuerdo a cuan típico de esa categoría es el caso que cada uno indica. Se elige como caso más representativo el más típico.

3.- Evaluar el grado de similitud entre el nuevo caso y el caso representativo elegido: la evaluación es realizada a través de un proceso de confrontación, guiado por:

- Un modelo del caso que se está tratando de clasificar.
- Asociaciones entre los componentes funcionales y su implementación en los casos.

4.- Reformular la hipótesis: si la confrontación no es exitosa, PROTOS utiliza las características que no confrontan para formar una mejor hipótesis, a través de un link diferencia que registre una de dichas características. Este link indica un nuevo caso que será usado como un caso representativo. Este caso puede pertenecer a la misma o a una categoría distinta. Por ejemplo, dado el caso de entrada:

Característica 1 (valor 1)
Característica 2 (valor 2)

Edad (más de 65 años)

Característica n (valor n)

suponiendo que un link recordativo nos lleva a elegir la categoría A y en la confrontación con el caso más típico de dicha categoría, se observan diferencias en la característica edad, PROTOS elige un link diferencia etiquetado con "edad (más de 65 años)" que nos lleva a un nuevo caso que pertenece a la categoría B. Si la confrontación con este caso es exitosa, PROTOS asigna la categoría B al caso de entrada.

Cuando no haya más links diferencia útiles, se vuelve al paso 2 para seleccionar otro caso representativo a través de otro link prototipo. Si no existen más de estos links, se elige otra categoría del conjunto de categorías probables.

Para la adquisición de conocimiento basada en casos, PROTOS cuenta con un experto que interviene cuando el sistema no puede clasificar un caso o cuando clasifica un caso en una categoría incorrecta. En esas situaciones, a través de un diálogo entre el experto y PROTOS,

este refina su conocimiento funcional relacionado al nuevo caso. Si luego, el caso es clasificado correctamente, se instala un nuevo link diferencia entre el caso representativo que llevo al error y el caso representativo que llevo a la categoría correcta, para que el mismo error no ocurra en el futuro.

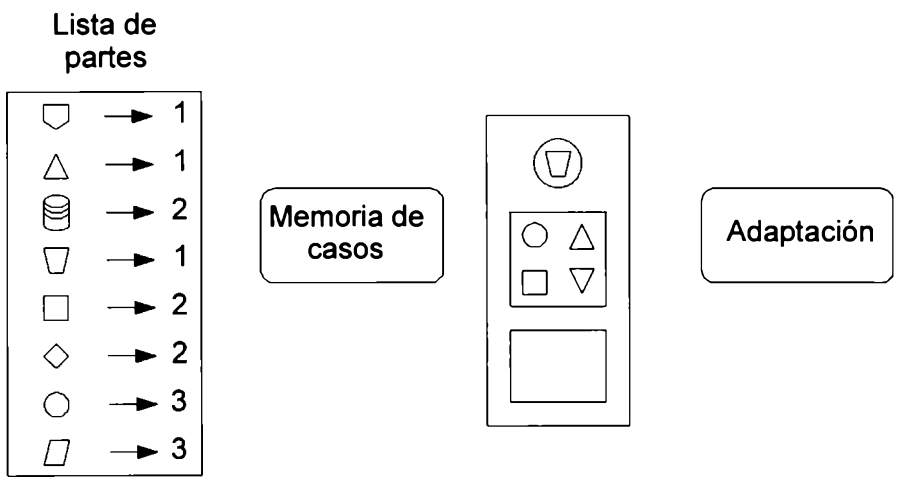
3.6.6) CLAVIER

Este sistema (desarrollado por Mark, Hennessy, Hinkle y Barletta [27]), configura el espacio de partes de un objeto que van a ser esterilizadas en un autoclave. Las partes están agrupadas en piezas.

CLAVIER, recibe como entrada una lista de partes con una prioridad para cada una, y da como resultado la configuración de la carga del autoclave, incluyendo la mayor cantidad posible de partes de la lista de entrada.

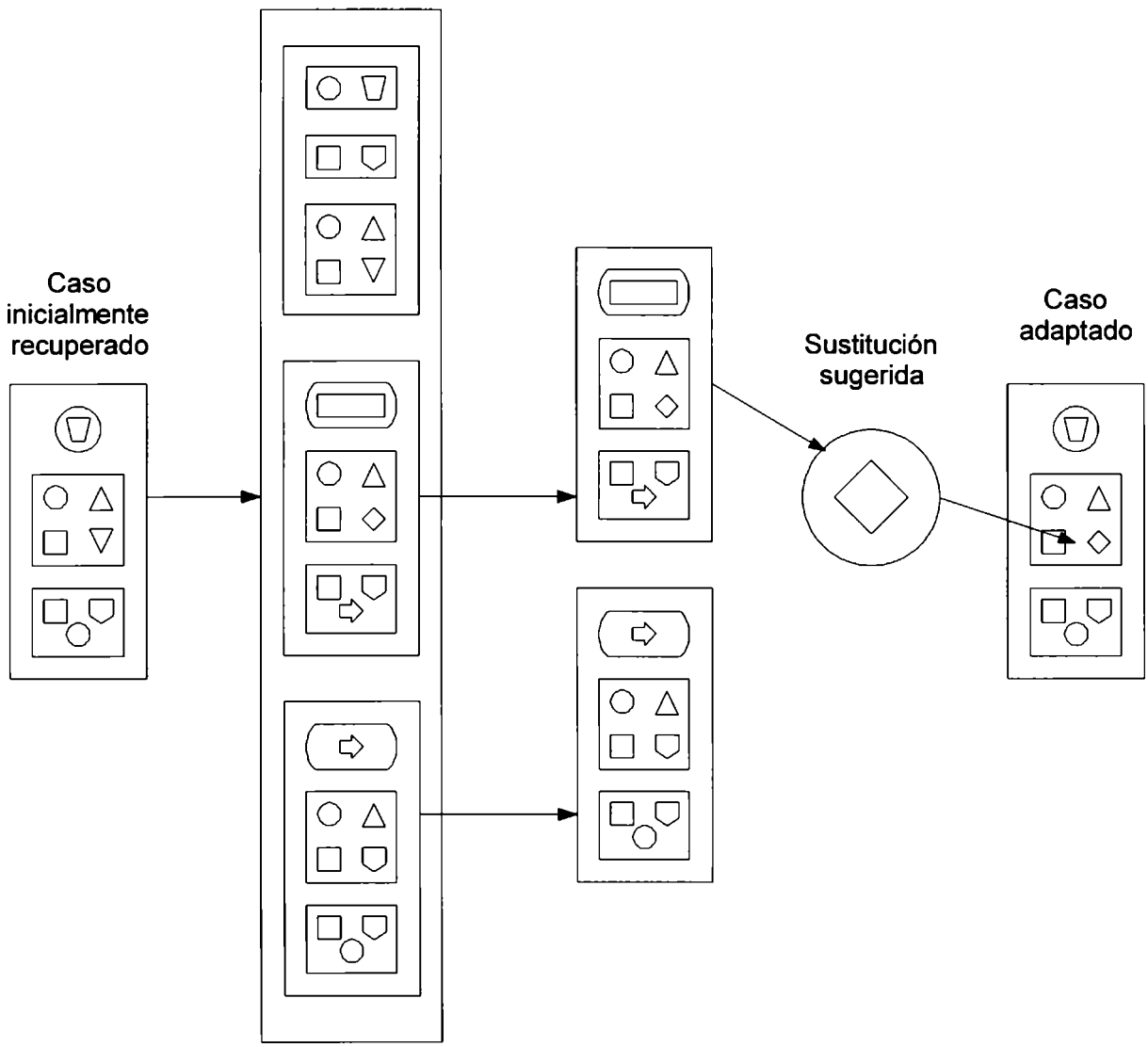
Cuenta con una librería de casos con configuraciones de partes en un autoclave, indexados por las partes que ellos incluyen.

CLAVIER recupera un caso, eligiendo el que incluya la mayor cantidad de partes de más alta prioridad. Este caso será modificado (si es necesario) a través de un proceso de adaptación. Por ejemplo:



El proceso de adaptación está basado en casos. Generalmente, alguna de las partes del caso recuperado no confronta con la especificación del problema. La adaptación comienza buscando en la librería de casos, casos con una pieza similar a la pieza que contiene la parte que no confronta en el caso recuperado. Para esto es necesario un paso previo llamado determinación del contexto, que especifica que buscar en la librería de casos, de acuerdo a dos parámetros: contexto global (contexto en el que está ubicada la pieza) y contexto local (descripción de la pieza que no confronta).

El paso siguiente es seleccionar uno de los casos sugeridos por la búsqueda anterior. Para esto, primero se reduce la cantidad de sugerencias, eligiendo aquellas donde la pieza en cuestión está en la misma ubicación que en el caso inicialmente recuperado. Se elige como sugerencia final, el caso que tiene, en la posición de la parte que no confronta, la parte de más alta prioridad de la lista de entrada. Esta es usada como sustitución de la parte en cuestión. El proceso de adaptación del ejemplo anterior es:



3.6.7) Battle Planer

Es un sistema de ayuda que juega el rol de un entrenador para un planeador de batallas (desarrollado por Goodman [27]).

Los sistemas de ayuda son sistemas interactivos que ayudan a la gente a solucionar problemas o que enseñan acerca de un dominio.

El usuario le describe, al sistema, la situación de una batalla y le da una supuesta solución. El sistema recupera los casos que usan un plan similar (dividiéndolos en exitosos y fallidos) y extrae información de ellos: comentarios sobre opciones y variaciones que fueron intentados en casos anteriores y un análisis comparativo. El usuario usa los casos y la información extraída para criticar su solución. Si la solución es inadecuada, el usuario propone una nueva y el proceso se repite. Por ejemplo, dada la siguiente situación, en la que Battle Planer toma la posición del defensor:

Situación	Atacante	Defensor
Nacionalidad	Soviéticos	Norteamericanos
Hombres	3700	1100
Tanques	84	64
Misión	Tomar la colina	Mantener territorio
Método	Asalto frontal	Línea defensiva estática

Se recuperan 9 casos, en los cuales el vencedor siempre fue el atacante y brinda la siguiente variantes que fueron tratadas en casos previos:

* En una batalla, el asalto rápido llevó a una victoria importante.
* En otras dos batallas, el retardo de acciones llevó a una segunda defensa exitosa.

Análisis comparativo: los siguientes factores favorecen la victoria del atacante:

- El defensor carece de reservas.
- El defensor carece de un frente "profundo".

Con estos datos el usuario genera una nueva situación:

Situación	Atacante	Defensor
Nacionalidad	Soviéticos	Norteamericanos
Misión	Tomar la colina	Retardar acciones
Método	Asalto frontal	Defensa en profundidad

3.7) Comparación y conclusión de razonamiento basado en casos

A partir del análisis de los métodos descriptos anteriormente, creímos útil mostrar como cada uno de los métodos trata las características relevantes de razonamiento basado en casos. La idea del cuadro es guiar al usuario de este survey a elegir el método que mejor se adecue a sus necesidades.

* Métodos que solucionan problemas vs métodos interpretativos: Los casos utilizados para solucionar problemas sugieren soluciones que luego serán adaptadas y advierten del peligro que ocurra una falla. Solucionar problemas basándose en casos reduce la cantidad de inferencias necesarias para construir la solución. Abarca tareas de diagnóstico, diseño y planificación. Ejemplos de estos son: CHEF, un sistema que planifica comidas y CASEY, un sistema que diagnostica desórdenes del corazón.

  El razonamiento interpretativo basado en casos es el proceso de evaluar situaciones o soluciones en el contexto de experiencias previas. Los casos proveen una base útil para interpretar nuevas situaciones, porque generalmente no se tiene el conocimiento necesario para utilizar los métodos computacionales conocidos. Abarca tareas de justificación, clasificación y proyección. Ejemplos de esto son: HYPO, un sistema que justifica una posición en un pleito dando un argumento y Battle Planer, un sistema que proyecta los efectos de un estrategia en un batalla.

* Todos los sistemas utilizan los casos para proponer soluciones o interpretaciones, pero algunos también los utilizan para otras tareas. Por ejemplo:
  - CHEF, JULIA y CLAVIER usan otros casos, además del inicialmente recuperado, para indicar adaptaciones.
  - HYPO utiliza casos opuestos a su posición como respuestas y casos a favor de su posición para refutarla. Además, en HYPO, los casos pueden ser utilizados para evaluar un argumento o para crear casos hipotéticos que lo evalúen.
  - PROTOS utiliza casos para proponer soluciones alternativas a la sugerida por el caso inicial.

* La anticipación de fallas es un mecanismo para prever los posibles errores que pueden aparecer en las soluciones. El proceso de anticipación de fallas es intencional cuando

se tiene un procedimiento particular para encontrar casos que indiquen una posible falla. Si la posibilidad de la falla aparece, como un efecto lateral, durante el proceso que encuentra la solución, la anticipación de fallas es no intencional.

Como ejemplo de anticipación de fallas intencional se encuentra CHEF, que realiza para esto un paso adicional, antes de comenzar la búsqueda de la solución. JULIA, como ejemplo de no intencional, cuando encuentra un caso que indica una falla, durante el proceso de búsqueda de la solución, examina si esta puede suceder en la solución corriente y si así fuera, toma las acciones adecuadas para evitarla.

* Algunos sistemas no sólo tienen el conocimiento dado por los casos y los índices, sino también otro tipo de conocimiento adicional.

Por ejemplo, CHEF contiene los críticos de objetos y la estructura TOP, que contiene las estrategias de reparación; CASEY usa reglas basadas en el mismo modelo causal utilizado por el programa que diagnostica fallas del corazón.

* Algunos programas indexan teniendo en cuenta sólo la especificación del problema, como por ejemplo JULIA; mientras que en otros se usa información más elaborada como método de indexación, como por ejemplo PROTOS que indexa a través de los links prototipos y los links diferencia.

* Algunos sistemas realizan una validación de la solución o interpretación creada.

CHEF corre el plan en un simulador para comparar los resultados obtenidos del simulador con los resultados esperados.

Otros sistemas cuentan con una persona quien se encarga de evaluar la solución; por ejemplo JULIA y PROTOS.

	Solucionar problemas o interpretar	Usos adicionales de los casos	¿Anticipa fallas?	Conocimiento adicional	¿Por qué indexan?	¿Cómo validan?
CHEF	Soluciona problemas. Planea	Para indicar estrategias de adaptación	Si, en el paso de anticipación de fallas	Críticos de objetos y las estrategias de reparación de la TOP	Por los objetivos que satisfacen y por los predictores de fallas	A través de un simulador

	Solucionar problemas o interpretar	Usos adicionales de los casos	¿Anticipa fallas?	Conocimiento adicional	¿Por qué indexan?	¿Cómo validan?
CASEY	Soluciona problemas. Diagnostica	———	No	Reglas basadas en un modelo causal	Por síntomas, signos y estados de diagnóstico del paciente	———
PROTOS	Soluciona problemas. Diagnostica	Para proponer soluciones alternativas	Si, a través de los links diferencias	Conocimiento funcional y modelos de las categorías	Por los links prototipos y por los links diferencias	A través de un experto
Battle Planer	Interpreta. Proyecta	———	No, porque la solución ya está dada	Un modelo del dominio	Por características derivadas	El sistema permite que el usuario evalue la solución
CLAVIER	Soluciona problemas. Diseña	Para realizar adaptaciones	No	———	Por las partes incluidas en el autoclave	———
HYPO	Interpreta. Justifica	Para dar respuestas y refutaciones y para evaluar	Si, buscando respuestas	Dimensiones	Por las dimensiones	A través de casos hipotéticos o casos reales

	Solucionar problemas o interpretar	Usos adicionales de los casos	¿Anticipa fallas?	Conocimiento adicional	¿Por qué indexan?	¿Cómo validan?
JULIA	Soluciona problemas. Diseña	Para adaptar partes de la solución	Si, cuando se recuerda un caso fallido	Prototipos de objetos, heurísticas de propósito especial y heurísticas para determinar la importancia de los componentes	Por la especificación del problema	A través de un cliente

4

Aprendizaje a partir de observaciones y descubrimientos

4.1) Introducción

En aprendizaje a partir de observaciones y descubrimiento (también llamado generalización descriptiva) el objetivo es buscar regularidades y reglas generales explicando las observaciones: descripciones de hechos, fenómenos u objetos que, a diferencia de los ejemplos, no especifican el hecho, fenómeno u objeto que describe [12].

Esta forma de aprendizaje es no supervisado, porque no recibe ningún beneficio de un instructor, por lo que este enfoque es el que hace más inferencias (las demás cuentan con la ayuda de un instructor).

Como las observaciones pueden abarcar varios conceptos, este tipo de aprendizaje está más relacionado a aprendizaje de múltiples conceptos que a aprendizaje de un único concepto.

De acuerdo al grado de interacción con el ambiente, puede ser que el sistema trabaje sobre observaciones recibidas o que el sistema perturbe el ambiente y observe los resultados de sus perturbaciones.

4.2) El uso de las heurísticas en aprendizaje por descubrimiento

Para que un programa en IA realice una tarea compleja, necesita un gran cuerpo de conocimiento experto, formado no sólo por hechos, sino también por heurísticas [18] (reglas de juzgamiento informales, que guían al experto) las cuales son difíciles de verbalizar.

El problema está en la lentitud con que el experto transfiere el conocimiento. Una solución para esto es exponer el programa al ambiente, permitiéndole descubrir el nuevo conocimiento por si mismo, sin considerar al experto.

Un dominio puede cambiar por la introducción de nuevos mecanismos, técnicas, teorías, paradigmas o fenómenos observables. Como las heurísticas tienen un dominio de relevancia, cuando un dominio cambia o surge uno nuevo, también cambia la utilidad de la heurística. Por lo tanto, si se tiene un conjunto de heurísticas de guía fijo, la capacidad de descubrir nuevo conocimiento declina con el tiempo. Entonces, en un determinado momento, puede valer la pena detener el proceso de búsqueda de nuevo conocimiento para que el programa se dedique a encontrar un nuevo conjunto de heurísticas. A partir de asumir que las heurísticas son como cualquier campo de conocimiento, es posible emplear un cuerpo de heurísticas para descubrir nuevas o modificar existentes.

También con el tiempo, el nivel del conocimiento puede cambiar, por lo tanto la representación del conocimiento puede resultar inadecuada. Lo primero que se intenta para solucionar esto, es modificar la representación existente agregando nuevos conceptos y/o predicados o definiendo propiedades para los conceptos recientemente descubiertos. Si aún la representación resulta inadecuada, se puede intentar cambiar a una nueva.

Se necesita un cuerpo de meta-heurísticas que indiquen cuando es necesario cambiar las heurísticas de guía y/o cambiar la representación.

4.3) Clasificación

Los sistemas en este enfoque pueden ser divididos en [10]:

- Sistemas de formación de teorías.
- Sistemas de agrupamiento conceptual.
- Sistemas que descubren patrones en los datos.

4.3.1) Formación de teoría

Formación de teoría involucra la elección del dominio, a ser expresado en un framework conceptual, y encontrar una expresión, en el framework, para explicar o definir el dominio. O sea, establecer nuevos conceptos o teorías caracterizando una colección de entidades.

4.3.1.1) Métodos de formación de teoría

A continuación se describe el método AM y varias aproximaciones al proceso llamado síntesis de programas.

4.3.1.1.1) AM

AM [18] es un programa que aplica aprendizaje por descubrimiento para modelar un aspecto de matemática elemental (teoría de conjuntos), desarrollando nuevos conceptos bajo la guía de un gran cuerpo de heurísticas. Cada concepto es representado por una estructura tipo frame con slots. Por ejemplo:

NOMBRE: primo
ES-UN: conjunto
EJEMPLOS:
  Ejemplos extremos: 2,3.

VALIDEZ: 800
ORIGEN: Divisores-de

AM utiliza una estrategia top-down. Inicialmente tiene 115 conceptos con la mayoría de sus slots en blanco. Tiene, también, una "agenda" de tareas a realizar. Cada una de estas tareas consiste en llenar un slot de un concepto. AM repetitivamente selecciona tareas de la agenda y las ejecuta, lo cual puede llevar a descubrir conceptos interesantes. Luego, las tareas

encargadas de completar la información (slots) de estos nuevos conceptos son puestas en la agenda. Por ejemplo, dada la tarea:

llenar el slots "ejemplo extremos" del concepto "divisores-de"

se considero ejemplos de números con dos divisores, que luego dio origen al concepto "primo".

AM usa un cuerpo de 250 heurísticas de guía para seleccionar un slot de un concepto a explorar, para obtener información acerca de un slot, para notar relaciones entre conceptos conocidos, para definir conceptos prometedores y para estimar cuan interesante es un concepto.

AM también incursionó en teoría de números, pero bajo la guía de heurísticas de teoría de conjuntos. La falta de heurísticas propias de teoría de números, lleva a definir muchos conceptos inútiles.

4.3.1.1.2) Síntesis de programas

Muchos de los esfuerzos de formación de teoría están dedicados a síntesis de programas: consiste en encontrar un programa, en un lenguaje de programación, que satisfaga un conjunto de datos de asociaciones entrada-salida.

A continuación se presentan 3 aproximaciones a síntesis de programas [19]:

Primera aproximación:

Dada una formulación declarativa de un problema, formada por la formulación de la clase de problema y los datos del problema específico, el sistema transforma ese conocimiento declarativo a una forma procedural, que le permita definir un proceso para generar soluciones candidatas. Estas soluciones serán testeadas sobre el problema específico hasta encontrar una que lo solucione.

Para generar soluciones candidatas, es necesario definir el lenguaje en términos de una gramática generativa. Luego, de acuerdo a la gramática, a cada programa en el lenguaje, se le puede asignar una descripción estructural.

El comportamiento del generador de soluciones candidatas puede ser representado por una árbol de búsqueda AND-OR, donde los nodos OR son nodos de selección de reglas de la gramática y los nodos AND son nodos de aplicación de reglas. Por lo tanto, un programa candidato surge de un conjunto de selecciones de reglas de la gramática. Cada selección es hecha de manera tal que se generen primero los programas candidatos más simples, para luego intentar con los más complejos si los primeros no fueron solución. La simplicidad de los

programas está determinada por los siguientes tres conceptos: complejidad de un programa, esfuerzo de búsqueda y sensitividad al cambio de una regla.

Segunda aproximación:

En esta aproximación se va a utilizar una modelo matemático del espacio de programas. Este modelo es una modificación del álgebra de relaciones; en él los términos están parcialmente ordenados por la implicación, esto permite dirigir la búsqueda en dicho espacio.

Existe una correspondencia entre un programa candidato y una expresión del álgebra de relaciones, de la que se puede obtener la correspondiente matriz de relación.

Por otro lado, la matriz de relación del programa objetivo puede ser obtenida a partir de los datos específicos del problema (las asociaciones).

El proceso que encuentra la solución está basado en una estrategia de generar y testear. Para cada generación, el método utiliza el estado corriente del problema, formado por:

- La expresión relacional que representa al problema candidato en el modelo algebraico.
- La "distancia" entre el programa candidato y el programa objetivo (basada en sus matrices de relación).

La idea del método es "navegar" en el espacio de programas, sucesivamente reduciendo la distancia entre el programa candidato y el programa objetivo, donde los posibles "movimientos" están dirigidos por la estructura implicacional del modelo algebraico (la inclusión entre las matrices de relación).

Para utilizar este método es necesario tener un modelo del espacio de programas y las propiedades útiles de tal modelo.

Tercera aproximación:

En esta aproximación se aumenta el razonamiento sobre el espacio de datos (las asociaciones). La idea es razonar con el conjunto de programas candidatos. Un programa es candidato cuando se espera que él produzca los caminos computacionales para construir todas las asociaciones. Para esto es necesario una buena representación de las acciones computacionales disponibles (condiciones y efectos).

Existen dos métodos alternativos en esta aproximación: el método combinación y el método eliminación.

En **el método combinación**, dada una asociación, el proceso genera esquemas de programas candidatos (a partir de la gramática) e intenta instanciarlos para obtener un programa que produzca un camino computacional para dicha asociación. Luego se testea dicho programa sobre las asociaciones restantes y se repite el proceso para alguna de las asociaciones no satisfechas, hasta que todas lo sean.

Como resultado se obtiene un conjunto de programas que deben ser combinados para satisfacer todas las asociaciones. Para solucionar este problema de objetivos conjuntivos se utiliza la aproximación modificación de programas: dado un programa que satisfaga algunas de las asociaciones, este es modificado para satisfacer las demás.

El método de eliminación es un caso especial de la aproximación de generalización para solucionar objetivos conjuntivos. Para una asociación, el proceso encuentra todos los programas que la satisfacen que estén debajo de un techo de complejidad dado (este techo de complejidad está determinado por la cantidad de reglas de gramática aplicadas para formar el programa). Dicho conjunto de programas es chequeado sobre la segunda asociación, eliminando los que no la satisfagan y así con el resto de las asociaciones.

Si en alguna iteración, no existen programas que satisfagan una asociación, se relaja la condición de complejidad y se buscan nuevos programas.

4.3.2) Agrupamiento conceptual

Agrupamiento conceptual es una aproximación al problema de crear una clasificación, que consiste en agrupar los objetos dados en clases conceptualmente simples.

Para construir una clasificación útil y significativa es necesario contar con conocimiento del ambiente: restricciones del problema, propiedades de los atributos, criterios para evaluar la calidad de agrupamiento, el objetivo general de la clasificación (que permite identificar y derivar descriptores relevantes), etc.

Dos aproximaciones a este problema son [20]:

- Discriminación repetida: reduce el problema de construir una clasificación a una secuencia de problemas de adquisición de conceptos.
- Atributos de clasificación: intenta encontrar uno o más atributos de clasificación cuyo conjunto de valores pueda ser dividido en rangos que definen clases individuales. Estos atributos pueden estar entre los atributos dados o ser derivados por inferencias, a partir de los atributos iniciales, dirigidas por el objetivo.

4.3.2.1) Métodos de agrupamiento conceptual

A continuación se describe el método CLUSTER/2, un ejemplo de la aproximación discriminación repetida.

4.3.2.1.1) CLUSTER/2

CLUSTER/2 [21] es un método de agrupamiento conjuntivo conceptual, que determina una jerarquía de clases caracterizando un conjunto de objetos; donde cada objeto es descripto por los valores de un conjunto de atributos predefinidos.

El algoritmo de CLUSTER/2 para construir la jerarquía de clases consta de 2 módulos:

1) El módulo de agrupamiento.
2) El módulo de construcción de la jerarquía.

Los dos módulos utilizan un "criterio de calidad de agrupamiento", especificado a través de una evaluación funcional lexicográfica, donde los 2 criterios más importantes son:

* El criterio de "dispersidad": evalúa el grado en que las descripciones de las clases generalizan los objetos observados.
* El criterio de simplicidad de las descripciones de las clases.

El **módulo de agrupamiento**, dado una colección de objetos y una cantidad de clase sugeridas, obtiene un agrupamiento disyunto (un conjunto de clases disyuntas que describen los objetos) que optimiza el criterio de calidad de agrupamiento.

El algoritmo aplicado por este módulo, construye sucesivos agrupamientos disyuntos, hasta que se satisface el criterio de parada.

En cada iteración, para cada clase, se elige un objeto (semilla) con algún criterio, con el que se genera un "**star**"[10] : conjunto de las posibles descripciones (llamadas complejos) de la clase.

---

[10] El procedimiento para generar un star para una clase C , dada una semilla e y un conjunto de objetos E (semillas de las demás clases) es [20]:

Para cada objeto e_j de E se extienden los selectores de e, es decir se extiende el rango de valores del selector sin cubrir el valor del mismo selector en e_j (definido para selectores con intersección vacia), para formar un conjunto S. En cada paso, si un elemento del conjunto S tiene intersección no vacia con e_j, este es multiplicado por los selectores extendidos (de dicho paso) para no cubrirlo. El star resultante es el último conjunto S, después de haber tratado a cada elemento de E.

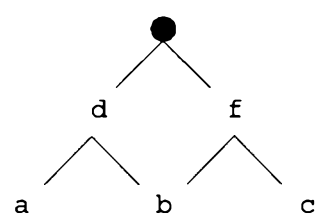
Por lo tanto, cada combinación, tomando un complejo de cada star (modificado, si es necesario, para que sea disyunto), es un agrupamiento disyunto posible. El agrupamiento disyunto elegido, en cada iteración, es el mejor de acuerdo al criterio de calidad de agrupamiento.

El criterio de parada está dado por: una cantidad fija de iteraciones (iteraciones base) y una cantidad de iteraciones adicionales (iteraciones de prueba), que son hechas después que es encontrado un agrupamiento mejor que los anteriores.

Por ejemplo, se quiere agrupar en dos clases el siguiente conjunto de eventos:

Evento	x1	x2	x3	x4
e1	0	a	0	1
e2	0	b	0	0
e3	0	c	1	2
e4	1	a	0	2
e5	1	c	1	1
e6	2	a	1	0
e7	2	b	0	1
e8	2	b	1	2
e9	2	c	0	0
e10	2	c	2	2

donde las variables x1 y x3 son lineales con los valores 0, 1 y 2, la variable x4 es nominal con los valores 0, 1 y 2 y la variable x2 tiene la siguiente estructura:



se elige e1 y e2 como semillas de las clases C1 y C2 respectivamente y los star resultante son:

star 1 (e1|e2) ={[x2 = a], [x4 = 1$\vee$2]}

star 2 (e2|e1) ={[x2 = f], [x4 = 0$\vee$2]}

la combinación de complejos elegida por LEF podría ser:

Complejo 1: [x2 = a]

Complejo 2: [x2 = f]

por lo tanto los objetos pertenecen a las siguientes clases:

Clase 1: {e1, e4, e6}

Clase 2: {e2, e3, e5, e7, e8, e9, e10}

se elige un objeto de cada conjunto como nueva semilla y se continúa de la misma manera hasta satisfacer el criterio de parada.

El **módulo de construcción de la jerarquía** construye la jerarquía de clases. Dado el conjunto de objetos observados, invoca iterativamente al módulo anterior, variando la cantidad de clases sugeridas, para determinar la mejor división de clases, cada una de estas estará representada por un nodo de la jerarquía.

Para cada nodo, el mismo proceso se repite recursivamente, así descendiendo en la construcción de la jerarquía, mientras se cumpla el "criterio de desarrollo continuo". Este criterio requiere que las descripciones en un nivel sean mejores que las del nivel anterior, de acuerdo al criterio de calidad de agrupamiento.

4.3.3) Descubrir patrones en los datos

Se trata se descubrir, a partir de observaciones dadas, las posibles relaciones, patrones o regularidades que existan en ellos.

4.3.3.1) Métodos de descubrimiento de patrones

A continuación se describen 4 métodos relacionados al descubrimiento científico: BACON-6, GLAUBER, STALH y DALTON; y un método de aplicación general: UNIMEM.

4.3.3.1.1) BACON-6

A partir de variables dependientes e independientes, con los valores que estas últimas pueden tomar, BACON-6 [22] descubre leyes empíricas de naturaleza cuantitativa, relacionando esas variables. Por ejemplo, la entrada a BACON-6 puede ser las siguientes variables independientes:

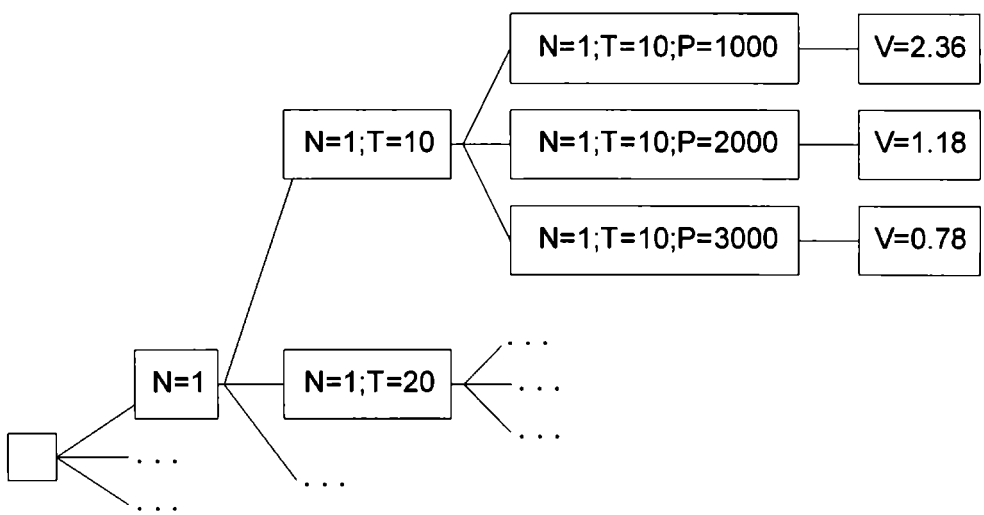
N (cantidad de gas) con los siguientes valores: {1, 2, 3}

T (temperatura del gas) con los siguientes valores: {10, 20, 30}

P (presión del gas) con los siguientes valores: {1000, 2000, 3000}

y la variable dependiente V (volumen del gas).

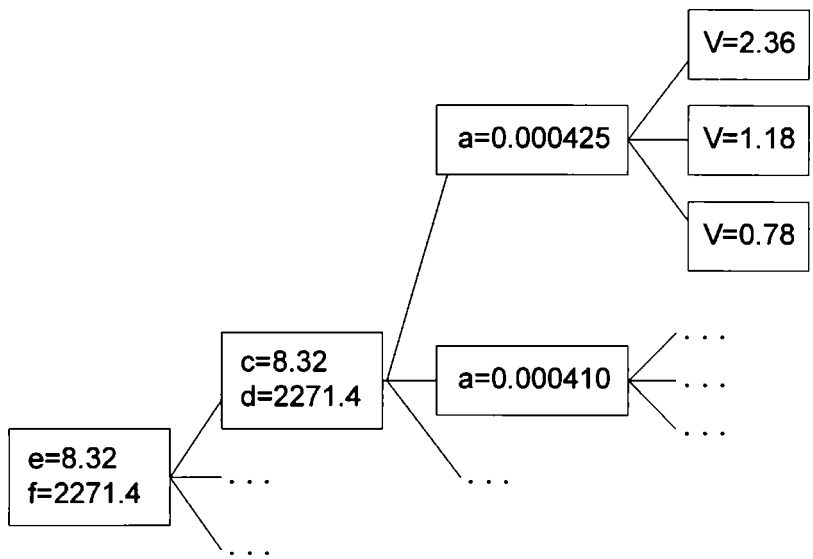
BACON-6 realiza búsqueda en dos espacios de problemas distintos: el espacio de datos y el espacio de leyes. La búsqueda en el espacio de datos genera todas las combinaciones posibles de los valores de las variables independientes, y observa, para cada combinación, los valores de los términos dependientes. Esta es una búsqueda depth-first y se debe usar backtraking para que todas las combinaciones posibles sean generadas. El espacio de datos generado con las variables del ejemplo es:



La búsqueda en el segundo espacio está embebido en el primer espacio. Dada la información acerca de las formas que pueden tomar las leyes admisibles (por ejemplo Y = a*x + b, donde y es la variables dependiente y x es la variable independiente), intenta encontrar la forma y los parámetros (a y b en el ejemplo anterior) que mejor predicen los datos observados, dados por las combinaciones anteriores. Para esto, BACON-6 realiza una búsqueda en el

espacio de parámetros, partiendo de un conjunto de estados iniciales[11], genera un nuevo conjunto de estados aplicando un operador que modifica los valores de los estados anteriores sumando y restando un número (por ejemplo, partiendo de 0.5 y reduciendo este valor en cada uno de los siguientes pasos). En cada paso, selecciona los mejores n estados para continuar la búsqueda. Un estado es mejor que otro cuando mejor predice los datos observados. La búsqueda termina cuando ningún nuevo estado mejora la predicción de los datos. En el ejemplo, dados los tres nodos terminales mostrados en la figura anterior y la forma de la ley y=a*x+b, el sistema encuentra que **a**=0.000425 y **b**=0.

BACON–6 representa la información a varios niveles de descripción. Las leyes encontradas a partir de los datos de un nivel, serán consideradas como datos en el nivel siguiente. El mismo proceso de búsqueda es aplicado recursivamente a través de los niveles. Por ejemplo, **a** es tomado como dato para el siguiente nivel:



A diferencia de las versiones anteriores de BACON, que sólo relacionan variables numéricas, BACON-6 tiene un método para definir propiedades intrínsecas que relacionan variables independientes nominales con variables numéricas dependientes. Esta propiedad toma valores numéricos y está asociada a las variables nominales. También se puede definir una propiedad especificando el "divisor común", si lo hay, entre los valores de la propiedad intrínseca.

Los valores de estas propiedades, como antes, son tomados como datos en un nivel más alto de descripción. Estos dos métodos adicionales en realidad transforman datos simbólicos en datos numéricos.

[11] Cada estado inicial está dado por la asignación de -1, 0 y 1 a cada parámetro.

4.3.3.1.2) GLAUBER

Según [22] GLAUBER es un sistema de descubrimiento que formula leyes empíricas cualitativas.

El sistema recibe como entrada un conjunto de hechos cualitativos, representados por una estructura tipo frame, donde cada hecho contiene un predicado y pares atributo/valor, por ejemplo en el dominio químico:

1- (tiene-calidad objeto{HCl} gusto{ácido})

2- (tiene-calidad objeto{HNO_3} gusto{ácido})

3- (tiene-calidad objeto{NaCl} gusto{salado})

4- (tiene-calidad objeto{KCl} gusto{salado})

5- (tiene-calidad objeto{$NaNO_3$} gusto{salado})

6- (tiene-calidad objeto{KNO_3} gusto{salado})

7- (tiene-calidad objeto{NaOH} gusto{amargo})

8- (tiene-calidad objeto{KOH} gusto{amargo})

9- (reacciona entradas{HNO_3 KOH} salidas{KNO_3})

En el proceso de descubrimiento cualitativo juega un rol importante definir clases y formular leyes. Por eso, el objetivo de GLAUBER es transformar los hechos de entrada en un conjunto de leyes cualitativas (con el mismo formato de los hechos originales, reemplazando valores por clases), y generar clases con una lista de miembros asociados.

Para esto, GLAUBER tiene dos operadores: un operador para formar leyes y otro para determinar su generalidad (cuantificación).

El primer operador, dado un conjunto de hechos con el mismo predicado y un valor de atributo común, obtiene un ley, donde el valor del atributo común es mantenido y se reemplazan los distintos valores de cada atributos por un nombre de clase (nuevo). Esta nueva clase agrupa todos estos valores distintos. La nueva clase debe reemplazar a todos sus miembros en los demás hechos, lo que puede llevar a generar una nueva ley. Por ejemplo, a partir de los hechos 3, 4, 5 y 6 se genera la siguiente ley:

$\forall$ sal (tiene-calidad objeto{sal} gusto{salado})

y la clase: sal {NaCl, KCl, $NaNO_3$, KNO_3}.

En cada hecho en que se reemplaza un valor por una clase, el segundo operador se encarga de determinar la generalidad de ella en dicho hecho, cuantificándolas universalmente,

si la mayoría de los hechos que cubre son hechos observados; en caso contrario son cuantificadas existencialmente. En el ejemplo, el conjunto de hechos y/o leyes resultantes después de aplicar los dos operadores es:

(tiene-calidad objeto{HCl} gusto{ácido})

(tiene-calidad objeto{HNO3} gusto{ácido})

(tiene-calidad objeto{NaOH} gusto{amargo})

(tiene-calidad objeto{KOH} gusto{amargo})

$\exists$ sal (reacciona entradas{HNO_3 KOH} salidas{sal})

$\forall$ sal (tiene-calidad objeto{sal} gusto{salado})

Estos dos operadores, además de ser aplicados a los hechos, pueden ser aplicados a las leyes de manera recursiva, tomando las clases como valores de atributos. En este caso, para poder aplicar el operador para formar leyes, no sólo se debe tener un predicado común y un argumento común, sino también tener la misma cuantificación. Por lo tanto en el ejemplo, el mismo proceso se repite para el último conjunto de hechos y/o leyes. Estos operadores son aplicados mientras sea posible.

La decisión sobre que leyes y que clases formar, está basada en la cantidad de repeticiones de un valor en un predicado a lo largo de todos los hechos.

4.3.3.1.3) STAHL

Este sistema (descripto en [22]), a partir de información cualitativa genera sentencias cualitativas, relacionadas a la estructura interna de sustancias químicas: dado un conjunto de reacciones químicas (por ejemplo reacción entrada(hidrógeno,oxígeno) salida (agua)), su objetivo es determinar los componentes de las sustancias participantes en dichas reacciones.

STAHL lleva a cabo una búsqueda depth-first, sin backtraking, a través del espacio de posibles componentes de una sustancia, dirigida por cinco operadores basados en heurísticas. El sistema obtiene inferencias mientras pueda aplicar operadores.

Estas cinco heurísticas son:

* Inferir composición: esta heurística le permite al sistema determinar los componentes de una sustancia a partir de síntesis y descomposición de reacciones; por ejemplo dada la reacción anterior, esta heurística infiere que el agua está compuesta por hidrógeno y oxígeno.
* Reducción: esta heurística propone eliminar una sustancia que está a ambos lados de una reacción. Con esta heurística se llega a versiones más simples de una reacción.

* Sustitución: esta heurística reemplaza una sustancia, en una reacción, por sus componentes, a partir de la inferencia, anteriormente hecha, de su composición.
* Identificación de componentes: cuando dos composiciones de una misma sustancia defieren sólo en un componente, esos dos componentes son considerados idénticos.
* Identificación de compuestos: como antes, si dos sustancias tiene la misma composición, ellas son consideradas idénticas.

Estas heurísticas interactúan de manera que la aplicación de una permite la aplicación de otra. Esta es la justificación de la aplicación de la heurística sustitución, porque, a pesar de que lleva a reacciones más complejas, permite la posterior aplicación de otras heurísticas, por ejemplo reducción.

El siguiente ejemplo ilustra la aplicación de algunas de las heurísticas, dada las siguientes entradas:

(reacción entrada {cal} salida {cal viva, dióxido de carbono})

(reacción entrada {cal viva, magnesio} salida {cal, óxido de magnesio})

primero el sistema aplica la heurística inferir composición a la primera reacción con lo que se determina los componentes de cal:

(componentes de {cal} son {cal viva, dióxido de carbono})

esta información permite la aplicación de la heurística sustitución a la segunda reacción:

(reacción entrada {cal viva, magnesio}
salida {cal viva, dióxido decarbono, óxido de magnesio})

esta reacción puede ser simplificada aplicando la heurística reducción eliminando el componente cal viva de ambos de la reacción:

(reacción entrada {magnesio} salida {dióxido de carbono, óxido de magnesio})

luego la heurística inferir composición permite determinar la composición de magnesio:

(componentes de {magnesio} son {dióxido de carbono, óxido de magnesio})

4.3.3.1.4) DALTON

DALTON [22] es un sistema de producción que descubre modelos estructurales. Dado un conjunto de reacciones e información sobre los componentes de las sustancias químicas que participan en las reacciones, como entrada, DALTON obtiene un modelo que especifica el número de moléculas y partículas para cada sustancia. Por ejemplo, se pide construir un modelo de la reacción del agua, dada la información que ella está compuesta por hidrógeno y oxígeno y que estas dos sustancias son elementos primitivos.

El sistema lleva a cabo una búsqueda depth-first a través del espacio de posibles modelos, tomando una reacción a la vez. Existen tres operadores para dirigir la búsqueda en tal espacio:

* Determinar el número de moléculas de una sustancia en una reacción.
* Determinar el número de partículas en una molécula en una reacción dada.
* Determinar también el número de partículas, pero de una forma más eficiente, exigiendo que se satisfaga la restricción de conservación. Esta restricción afirma que el número de partículas de una sustancia dada como entrada a una reacción debe mantenerse en la salida de dicha reacción.

Comenzando con un modelo abstracto, DALTON intenta determinar el número de moléculas y luego el número de partículas. Primero considera los modelos más simples (modelos que especifican una molécula por sustancia y una partícula por molécula). Cuando un modelo está totalmente instanciado, se verifica la restricción de conservación. Si la restricción no se cumple, realiza backtraking, intentando con los modelos más complejos. En el ejemplo, DALTON partiendo del modelo abstracto (H O $\rightarrow$ W) para el agua, elige la hipótesis más simple, determinando primero una molécula para hidrógeno, luego una molécula para oxígeno y luego una molécula para agua: ((H) (O) $\rightarrow$ (W)).

Luego, asume que se tiene una partícula por molécula generando el siguiente modelo: ((h) (o) $\rightarrow$ (W)).

Se intenta finalizar el modelo de manera tal que se satisfaga la restricción de conservación. El operador de conservación determina que la molécula de agua debe estar compuesta por una molécula de hidrógeno y una de oxígeno: ((h) (o) $\rightarrow$ (h o)).

DALTON agrega una restricción adicional, la cual exige que el modelo de una sustancia sea igual para toda reacción; esto restringe el espacio de búsqueda con los modelos previamente descubiertos.

Si a DALTON se le provee información adicional, esta le puede llevar a considerar alternativas que sino no hubiera considerado. Por ejemplo, si se agrega una heurística que

dice: "la combinación de volúmenes está relacionada al número de moléculas involucradas en una reacción", se puede obtener el siguiente modelo: ((h) (h) (o o) → (h o) (h o)).

Una vez que se encontró un modelo satisfactorio, este debe ser almacenado como una producción para su posterior uso.

Con algunas modificaciones mínimas, DALTON puede ser adaptado a otros dominios como física.

4.3.3.1.5) UNIMEM

UNIMEM [15] es un sistema de información inteligente que aprende a partir de una gran cantidad de hechos no estructurados acerca de un dominio, donde cada hecho es descripto en términos de un conjunto de características (pares propiedad/valor).

Los sistemas de información inteligentes adquieren múltiples conceptos, no sólo comparando hechos, sino también determinando cuales comparar. Estos deben satisfacer las siguientes propiedades:

* Generalizaciones pragmáticas: los conceptos describen lo que es usualmente verdadero, pero no necesariamente verdadero. Esta clase de generalizaciones da más poder y flexibilidad para representar lo que es posible aprender acerca de un dominio rico.
* Aprendizaje incremental.
* Tratar con gran cantidad de hechos.

UNIMEM utiliza una memoria basada en generalizaciones (GBM). Esta técnica permite determinar que conceptos aprender y la definición de esos conceptos.

La idea básica de GBM, es que un sistema de generalización cree una jerarquía de conceptos, registrando en memoria los conceptos y los hechos a partir de los cuales ellos fueron generalizados. Esta jerarquía de conceptos va desde los conceptos más generales a los más específicos. Por ejemplo la siguiente jerarquía representa algunos estados de E.E.U.U.:

Nodo 0

[Arizona Massachusetts NuevoMexico]

Nodo 1		
Industria	tipo	manufacturación (20)
Industria	tipo	turismo (-2)
Industria	tipo	agricultura (16)

Impuestos	rango	2:5 (14)
[ ]		

Nodo 5		
rentas	rango	3:4 (4)
Industria	tipo	minería (1)
gastos en educación	rango	3:3 (0)
[Utah]		

Nodo 7		
cantidad de crímenes	rango	5:5 (-1)
deuda del estado	rango	3:7 (1)
Industria	tipo	gubernamental (-1)
[Colorado Nevada]		

Organizar la memoria de esta manera permite un eficiente almacenamiento de la información, ya que las características de una generalización no tiene que ser repetida para cada una de sus instancias; también permite encontrar eficientemente en memoria las generalizaciones y las instancias relevantes, y sólo ellas, durante el procesamiento.

El proceso de mantener la GBM es relativamente simple: cuando un nuevo hecho es procesado, se busca el concepto más específico que lo describe. Se hace un chequeo entre las instancias ya almacenadas (para esta generalización) y la nueva instancia. Si ellas describen un nuevo concepto, él es creado; sino, la nueva instancia es almacenada como instancia del concepto que la describe, anteriormente encontrado. Dada la jerarquía anterior y el siguiente hecho:

Oregon		
gastos en educación	rango	3:3
cantidad de crímenes	rango	4:5
rentas	rango	3:4
Impuestos	rango	2:5
Industria	tipo	manufacturación
Industria	tipo	turismo
Industria	tipo	agricultura
deuda del estado	rango	5:8
tamaño del estado	rango	4:6

se obtiene la nueva jerarquía:

Nodo 0
[Arizona Massachusetts NuevoMexico]

Nodo 1

Industria	tipo	manufacturación (21)
Industria	tipo	turismo (-1)
Industria	tipo	agricultura (17)
Impuestos	rango	2:5 (15)

[ ]

Nodo 5

rentas	rango	3:4 (5)
Industria	tipo	minería (0)
gastos en educación	rango	3:3 (1)

[ ]

Nodo 7

cantidad de crímenes	rango	5:5 (-2)
deuda del estado	rango	3:7 (0)
Industria	tipo	gubernamental (-2)

[Colorado Nevada]

Nodo 9

cantidad de crímenes	rango	4:5 (0)
tamaño del estado	rango	4:6 (0)

[Utah Oregon]

Como Oregon comparte con Utah las características "cantidad de crímenes" y "tamaño del estado" se creó un nuevo nodo (Nodo 9) que los agrupa.

Como generalmente, los conceptos son creados generalizando sólo unos pocos hechos, ellos deben ser evaluados para prevenir que contengan información (características) extra que surgen de coincidencias. Para esto, una "confianza" es mantenida para cada característica de cada concepto. Un nuevo hecho modifica la confianza de las características del concepto que es relevante para él, de acuerdo a si la confirma o no. Por ejemplo, el hecho que representa a

Oregon, entre otras, disminuye la confianza de la característica "industria tipo minería" y aumenta la confianza de la característica "impuestos".

Cuando la confianza está por debajo de cierta tolerancia, la característica es eliminada. Y si demasiadas características han sido borradas de una generalización, ella es eliminada. Este esquema permite eliminar conceptos que fueron el resultado de coincidencias.

Este método también podría haber sido clasificado como un método de agrupamiento conceptual.

4.4) Comparación y conclusión

A partir del análisis de los métodos descriptos anteriormente, creímos útil mostrar como cada uno de los métodos trata las características relevantes de aprendizaje a partir de observaciones y descubrimientos. La idea del cuadro es guiar al usuario de este survey a elegir el método que mejor se adecue a sus necesidades.

* La mayoría de los programas son muy específicos para un dominio dado. Sería muy difícil realizar un sistema en este tipo de aprendizaje abstrayéndose de un dominio, porque la utilidad de un programa depende de la incorporación de conocimiento específico del ambiente.

  Los más específicos para un dominio, entre los programas nombrados, son: STAHL y DALTON, quienes trabajan en el dominio químico y AM, quien trabaja en teoría de conjuntos. Algunos pueden ser extendidos a dominios relacionados, pero perdiendo utilidad, como AM, que se puede extender a teoría de números, y DALTON, que se puede extender a la física.

  Entre los más generales puede nombrarse a CLUSTER/S y UNIMEM.

* Todo tipo de aprendizaje a partir de observaciones y descubrimientos necesita más información del ambiente (dominio específico) que las demás aproximaciones; esto se debe a la falta de ayuda de un instructor.

  Algunos programas incorporan el conocimiento del ambiente mediante heurísticas. Por ejemplo, AM incorpora heurísticas de teoría de conjuntos y STAHL contiene 5 operadores cada uno implementando una heurística proveniente de la química.

  CLUSTER/S utiliza información del ambiente para definir un criterio de calidad de agrupamiento y una red de dependencia de objetivos que le permite obtener un agrupamiento conceptual relevante al objetivo.

BACON–6 necesita conocer el conjunto de formatos admisibles de las leyes a descubrir.

* Algunos sistemas finalizan cuando llegan a un objetivo determinado. Por ejemplo, CLUSTER/S lo hace cuando descubre el agrupamiento conceptual óptimo.

Otros sistemas, en cambio, continúan descubriendo mientras puedan, por ejemplo AM aprende conceptos matemáticos mientras las heurísticas se lo permiten.

* Algunos programas utilizan los conceptos descubiertos para restringir el espacio de búsqueda de nuevos descubrimientos, como BACON–6 que restringe los formatos de leyes, suponiendo que si un formato fue útil en un contexto también lo será en contextos similares.

En otros programas, los conceptos aprendidos llevan a nuevos descubrimientos que sino no hubieran surgido, por ejemplo en GLAUBER una ley puede llevar a descubrir otras.

		¿Descubre mientras puede?	Específicos o generales	¿Como usa los descubrimientos hechos?	Conocimiento del ambiente	Estrategia de control
Agrupamiento conceptual	CLUSTER\S	Finaliza cuando descubre el agrupamiento conceptual óptimo	General	———	El criterio de calidad de agrupamiento y la red de dependencia de objetivos	Top-down
Descubrimiento de patrones	STAHL	Si. Descubre mientras los operadores se lo permitan	Específico del dominio químico	Una composición aprendida le puede permitir descubrir otra	Las heurísticas para los operadores de química	Bottom-up

		¿Descubre mientras puede?	Específicos o generales	¿Como usa los descubrimientos hechos?	Conocimiento del ambiente	Estrategia de control
Descubrimiento de patrones	DALTON	No. Finaliza con el descubrimiento de los modelos de las reacciones	Específico del dominio químico	El descubrimiento de un modelo de una sustancia en una reacción restringe el espacio de búsqueda de otros modelos	Se puede agregar conocimiento adicional para mejorar su comportamiento	Bottom-up
	GLAUBER	Si. Descubre leyes mientras los operadores se lo permiten	General. Pero fue diseñado basándose en descubrimientos químicos	Una ley aprendida le puede llevar a descubrir nuevas leyes	———	Bottom-up
	UNIMEM	Como es incremental, aprende mientras llegan observaciones	General	———	———	Bottom-up

		¿Descubre mientras puede?	Específicos o generales	¿Como usa los descu-brimientos hechos?	Conoci-miento del ambiente	Estrategia de control
Descubri-miento de patrones	BACON-6	Si. Aprende mientras pueda rela-cionar varia-bles depen-dientes e indepen-ientes	General. Pero fue di-señado ba-sánose en descubri-mientos quí-micos	El descubri-miento que un formato es útil en un contexto, restringe a usar ese for-mato en otro contexto	Los forma-tos admisi-bles de las leyes	Mixto. En el espacio de datos aplica una estrate-gia bottom-up y en el espacio de leyes top-down
Formación de teoría	AM	Si. Descu-bre nuevos conceptos mientras las heurísticas se lo permi-tan	Específico del dominio de teoría de conjuntos	El descubri-miento de un concep-to le puede permitir agregar nuevas ta-reas a la agenda	Las heurísti-cas de teo-ría de con-juntos	Top-down

5

Aprendizaje a partir de instrucción

5.1) Introducción

En este paradigma de aprendizaje, el sistema adquiere conocimiento a partir de un instructor externo, necesitando transformar el conocimiento de entrada a la representación interna. Además el nuevo conocimiento debe ser correctamente integrado al conocimiento existente, para un uso efectivo.

Este tipo de aprendizaje, es uno de los que más intervención necesita de una fuente externa; ya que el instructor debe organizar y presentar el conocimiento de forma tal que el sistema aumente su capacidad.

Aprendizaje a partir de instrucción, es el paradigma más comparable al aprendizaje usado en los métodos de educación formal.

5.2) Métodos de aprendizaje a partir de instrucción

A continuación se describen los métodos FOO y KLAUS dos ejemplos de este tipo de aprendizaje.

5.2.1) FOO

FOO [24] es un programa que implementa las reglas necesarias para operacionalizar una tarea: transformar la especificación de la tarea en un mecanismo computacional general para realizarla.

Este programa es aplicable a tareas de la forma: encontrar una secuencia de elementos, llamada camino, que satisface una criterio dado. Iterativamente, dado un camino parcial, se debe elegir una elemento para agregar al camino, hasta llegar a la solución. Por ejemplo, un juego de cartas donde existe un palo conductor (palo de la carta jugada por el primer participante), cada jugador juega una carta por turno y el jugador que en una mano juegue la carta más alta en el palo conductor suma los puntos de las cartas de la mano (sólo las cartas del 1 al 10 de cada palo suman puntos). El objetivo del sistema es evitar que un jugador P sume puntos.

El mecanismo computacional que FOO encuentra es un procedimiento de búsqueda heurístico. Este está representado por un grafo de flujo de datos, cuyos componentes deben ser instanciados para solucionar un problema particular. Algunos componentes son:

"Test solución": verifica cuando un camino es solución.

"Orden de caminos": define que caminos considerar primero.

FOO provee la representación del problema a operacionalizar y un conjunto de reglas de transformación de problemas, pero carece del conocimiento de control; este debe ser dado por el instructor: elegir la secuencia de reglas y a que componente aplicar cada una. Además, el instructor debe proveer conocimiento del dominio: conceptos utilizados en la tarea. Por ejemplo, dado el problema anterior, la representación utilizada por FOO es:

(evitar que P (tome puntos) (en la mano))

algunos de los conceptos dados por el instructor son: palo conductor, cartas con puntos, etc y algunos de las reglas de transformación de FOO son:

Regla 1: "Sustituir en la representación de un problema un concepto por su definición.

Regla 2: "Mover restricciones desde el componente "Test solución" al componente "Orden de caminos".

FOO, primero construye un procedimiento de búsqueda generar y testear (sin heurísticas) aplicando algunas de sus reglas para instanciar los componentes del grafo general. Este es un procedimiento de búsqueda exhaustiva que testea si todo camino generado es solución al problema. Para dicha instanciación existen varios pasos:

1.- Se debe reconocer que el problema es del tipo "encontrar una secuencia de elecciones satisfaciendo una condición". Si no es así, se debe reformular el problema. En el ejemplo, el subproblema tomar puntos en la mano es reformulado como:

Problema: sumar puntos en la mano.

Objeto: mano.

Secuencia de elecciones: secuencias de elecciones (de la mano).

2.- Se debe identificar el espacio de búsqueda (en función de la secuencia de elecciones): los puntos donde se hacen las elecciones, las alternativas para cada una y el test que debe satisfacer un camino solución. En el ejemplo, se identificaron los componentes:

Elecciones: cartas jugadas.

Conjunto de elecciones de un jugador: cartas legales.

Test de completitud: en la mano, debe existir una carta por jugador.

3.- Se debe reformular el criterio de búsqueda como una función de la secuencia de elecciones. En el ejemplo, se reformula tomar puntos en la mano en función de cartas jugadas:

([tengan puntos las cartas jugadas] y [la carta jugada por P es la mayor de las cartas jugadas en el palo conductor])

4.- Hacer el procedimiento ejecutable con los datos disponibles, es decir, que cada componente pueda ser evaluado con los datos disponibles. En el ejemplo, el componente "Conjunto de elecciones de un jugador" definido anteriormente como sus cartas legales no es evaluable, porque no se puede conocer cuales son las cartas de un oponente. Entonces, este componente es reformulado como:

Conjunto de elecciones de un jugador: cartas posibles.

donde cartas posibles son las cartas no jugadas.

El proceso anterior es transformado en un procedimiento más eficiente de búsqueda heurística incorporando heurísticas basadas en las propiedades del problema y el dominio de la tarea. La idea es incorporar las restricciones del problema a la búsqueda. La búsqueda puede mejorar de varias formas:

- Podar el espacio de búsqueda reduce la cantidad de alternativas a considerar, identificando y eliminando aquellos caminos parciales que no pueden ser extendidos para producir una solución. Por ejemplo, se pueden eliminar los caminos parciales en los que la carta de P no sea la más alta en el palo conductor, porque seguro que P no suma puntos.
- Ordenar la búsqueda, considerando primero aquellos caminos parciales que tienen mayor probabilidad de alcanzar la solución. Esto permite encontrar más rápido la

solución. Por ejemplo, se considerarán primero aquellos caminos parciales cuyas cartas sumen puntos.

- Compilar restricciones fuera de la búsqueda: algunos elementos del camino pueden ser determinados antes de comenzar la búsqueda si existen datos que así lo indiquen. Esto reduce la profundidad del espacio de búsqueda. Por ejemplo, si el propósito es determinar que carta debe jugar un jugador P para evitar que tome puntos, la información de las cartas que tiene P y las cartas que jugaron los predecesores de P puede ser especificada antes de comenzar la búsqueda.
- Unir elecciones equivalentes en elementos abstractos, eliminando los detalles de los elementos originales que son irrelevantes al criterio de búsqueda. Esto permite reducir el espacio de búsqueda. Por ejemplo, se puede eliminar el detalle del palo de la carta considerando los elementos abstractos "cartas que pertenecen al palo conductor" y "cartas que no pertenecen al palo conductor".

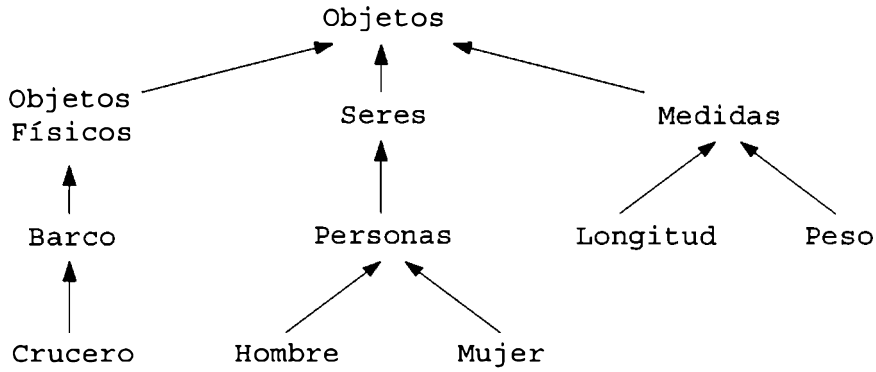
5.2.2) KLAUS

Los sistemas "KLAUS" [23] son sistemas que adquieren conocimiento acerca de un dominio específico y usan subsistemas de software externos para solucionar los problemas del usuario (por ejemplo: sistemas manejadores de bases de datos, generadores de reportes, etc.).

Un sistema KLAUS adquiere el conocimiento específico, necesario para sistemas inteligentes de manejo de información, por medio de diálogos interactivos en lenguaje natural.

La adquisición de conocimiento de KLAUS, no sólo consta de aprendizaje de conceptos, sino también de las construcciones lingüísticas necesarias para expresarlos.

NANOKLAUS es un prototipo de un sistema KLAUS, que mantiene conversaciones en inglés con el usuario. Comienza con un conjunto de reglas preprogramadas semánticas y sintácticas del inglés, un conjunto de conceptos iniciales que será extendido a medida que el sistema aprende y la jerarquía de clases de esos conceptos (jerarquía tipo "es-un", que mantiene información de los conceptos y sus clases). Por ejemplo, una jerarquía de clases que incluye los conceptos iniciales es:



Dentro del conjunto de reglas de NANOKLAUS, existen reglas específicas de propósito especial, para procesar los distintos tipos de sentencias de entrada. Por ejemplo, la siguiente regla:

<SENTENCIA> ⇒ <VERBO> <ARTÍCULO> <SUSTANTIVO CONOCIDO>
| (MOSTRAR <SUSTANTIVO CONOCIDO>)

permite procesar la sentencia de entrada:
Listar los barcos

Si la sentencia de entrada confronta la primera parte de la regla, entonces NANOKLAUS ejecuta MOSTRAR < SUSTANTIVO CONOCIDO > (en el ejemplo, lista todos las instancias del concepto barco, que existen en su base de datos).

También tiene patrones sintácticos para definir nuevos conceptos. Cuando un hecho de entrada (del usuario) confronta con uno de estos patrones, se llama al procedimiento de asimilación asociado al patrón. El procedimiento de asimilación se encarga de almacenar el hecho en la base de datos e insertar el concepto aprendido en la jerarquía de clases. Para esto, NANOKLAUS, debe hacer preguntas al usuario para obtener la información necesaria. Por ejemplo, un patrón sintáctico de NANOKLAUS es:

<SENTENCIA> ⇒ <UN> <NUEVA PALABRA> <ES> <UN> <SUSTANTIVO CONOCIDO>

a través de cual, se puede definir un nuevo concepto a partir del siguiente hecho de entrada:
Un carguero es un barco
el procedimiento de asimilación asociado a este patrón, agrega el concepto carguero a la jerarquía de conceptos y realiza pregunta al usuario para determinar su relación con las otras clases de objetos barco, por ejemplo realiza la siguiente pregunta:
¿Crucero es un subclase de carguero?

El esquema de representación de conocimiento está basado en lógica de primer orden. La jerarquía de clases del sistema, está representada por una estructura de tipo árbol.

Cada vez que un nuevo hecho es agregado, NANOKLAUS chequea para ver si es consistente con el conocimiento anterior. Pero no tiene forma de "desaprender" un hecho incorrecto aprendido anteriormente.

5.3) Comparación y conclusión

A partir del análisis de los métodos descriptos anteriormente, creímos útil mostrar como cada uno de los métodos trata las características relevantes de aprendizaje a partir de instrucción. La idea del cuadro es guiar al usuario de este survey a elegir el método que mejor se adecue a sus necesidades.

* El instructor puede proveer distintos tipos de conocimiento: conocimiento acerca de un dominio específico, conocimiento de control, etc.

  En FOO, el instructor debe proveer el conocimiento de control para la operacionalización y conocimiento del dominio.

  En un sistema KLAUS, el instructor ingresa sentencias: hace preguntas al sistema, responde preguntas del sistema e ingresa hechos.

* En KLAUS, tanto el usuario como el sistema, controlan la instrucción, por un lado, el instructor hace preguntas al sistema e ingresa nuevo conocimiento y por otro, el sistema le hace preguntas al instructor.

  En FOO, el sistema no influencia la instrucción.

* En KLAUS, el sistema y el instructor, mantienen un diálogo interactivo en alguna forma de lenguaje natural; mientras que en FOO, no existe diálogo (no está claro cómo se introduce el conocimiento de control).

* Todos los métodos descriptos pueden ser aplicados a cualquier dominio, con la aclaración de que FOO puede ser aplicado sólo a problemas de la forma "encontrar una secuencia de elementos que satisfacen un criterio dado".

	¿Que provee el instructor?	¿Quien controla la instrucción?	¿Como se realiza la instrucción?
FOO	Conocimiento de control y del dominio	El instructor	Por medio de un proceso de guía manual
KLAUS	Provee respuestas e ingresa hechos	El instructor y el sistema	Por medio de diálogos interactivos en lenguaje natural

6

Conclusión final

Este capitulo desarrolla una comparación de todas las estrategias descriptas anteriormente, comparando aspectos relevantes

- Cada estrategia necesita contar con diferente información de entrada:

  En aprendizaje a partir de ejemplos, es necesario tener una base de ejemplos apropiada donde cada uno está descripto por atributos y relaciones y la clase a la que ejemplifican. Dependiendo del método, se puede requerir que la base de ejemplos tenga ejemplos de varias clases, ejemplos positivos y negativos de una clase o sólo ejemplos positivos de una clase.

  En aprendizaje a partir de observación y descubrimiento, es necesario tener una base de observaciones donde cada una está descripta de forma similar a los ejemplos, con la diferencia que dicha descripción no incluye una clase (porque dada un observación no se sabe lo que esta describe).

  En aprendizaje por analogía, es necesario contar con un conjunto de experiencias pasadas, donde cada una contenga la información que permita resolver la nueva situación.

  En aprendizaje a partir de instrucción, cada uno de los métodos le pedirá al instructor la información que él necesite.

- En general la utilidad de una método de aprendizaje, no depende de la estrategia que aplique, pero pueden notarse algunos usos generales:

  En observación y descubrimiento los métodos son utilizados para:

  * Formación de teorías: a partir de un conjunto de observaciones, se genera una teoría que las explique.
  * Agrupamiento conceptual: a partir de observaciones que abarcan varios conceptos, encontrar un agrupamiento que refleje dichos conceptos.
  * Descubrimiento de patrones: descubre regularidades, patrones o relaciones que existan en las observaciones.

En aprendizaje a partir de ejemplos, los métodos son utilizados para aprender reglas y descripciones de conceptos.

En aprendizaje por analogía, generalmente se obtienen soluciones a problemas, basándose en experiencias pasadas.

- Un sistema de aprendizaje realiza una serie de transformaciones o inferencias, a partir de la información dada por un instructor o tomada del ambiente, obteniendo el conocimiento adquirido por el sistema.

  Cuando más inferencias hace el sistema, menos se necesita la ayuda de un instructor. Cada estrategia de aprendizaje define una relación entre el esfuerzo del sistema y el esfuerzo del instructor.

  En aprendizaje a partir de instrucción, el mayor esfuerzo está asignado al instructor, quien debe presentar y organizar la información de forma tal que el sistema aumente su conocimiento.

  Algunos sistemas de aprendizaje por analogía, requieren que el instructor provea la métrica de similitud necesaria para recuperar una base análoga adecuada. En cambio otros, requieren directamente que el instructor de el consejo análogo. Esta estrategia, comparándola con la anterior, requiere más esfuerzo del sistema y menos del instructor.

  En aprendizaje a partir de ejemplos, en el caso que la fuente sea el instructor (lo que comúnmente sucede), él debe generar la secuencia de ejemplos tratando de alcanzar, lo más rápidamente posible, el o los concepto/s objetivo/s.

  En aprendizaje a partir de observación y descubrimiento, todo el esfuerzo es realizado por el sistema, ya que este tipo de aprendizaje es no-supervisado (no cuenta con la ayuda de un instructor).

- Los métodos específicos son más eficientes y producen mejores soluciones que los generales, porque tienen incorporado heurísticas y conocimiento del dominio.

  Generalmente, los métodos de aprendizaje a partir de observación y descubrimiento son específicos, porque ellos deben disponer de conocimiento propio del dominio.

  En las demás estrategias pueden desarrollarse métodos que son aplicables a varios dominios, porque no es necesario tanto conocimiento específico. Estas estrategias también pueden aplicarse a un determinado dominio para lograr las mejoras dichas anteriormente.

	Información de entrada	Utilidad del método	Esfuerzo por parte del instructor	Específico vs general
Aprendizaje a partir de ejemplos	Ejemplos	Aprender reglas y descripciones de conceptos	Generar la secuencia de ejemplos	General
Aprendizaje por analogía	Problemas semejantes	Solucionar problemas	Presentar la métrica de similitud o el consejo análogo	General
Aprendizaje a partir de observación y descubrimiento	Observaciones	Generar teorías, descubrir agrupamientos conceptuales y descubrir patrones	Ninguno	Específico
Aprendizaje a partir de instrucción	Conocimiento aportado por el instructor	Aprender descripciones de conceptos, operacionalizar una tarea, etc	Presentar toda la información necesaria	General

PARTE II

Prototipo

7

Descripción y Análisis del Prototipo ID3

7.1) Introducción

Como segunda parte de nuestro proyecto deseamos aplicar sobre un dominio real algunos de los métodos descriptos anteriormente.

A partir de la inquietud del doctor Chappa de automatizar su sistema de diagnostico psiquiátrico, se comenzó con un análisis de su forma de trabajo. Se concretaron una serie de entrevistas con el objetivo de comprender totalmente su modalidad de trabajo. El psiquiatra utiliza procedimientos de autoinforme, cuyo empleo permite al paciente mismo evaluar su experiencia sintomática en una escala. Esta es una técnica de evaluación estructurada en el estudio de pacientes psiquiátricos. Este procedimiento consta de una serie de cuestionarios de respuestas forzadas que el paciente contesta en la primera entrevista. Cada cuestionario tiene como resultado un puntaje, que se desprende de un cálculo predefinido de las respuestas. Cada cuestionario está relacionado a un aspecto de la psiquiatría; el rango al que pertenece el puntaje determina el perfil del paciente en ese aspecto. Luego el psiquiatra realiza una serie de entrevistas personales para conocer más detalladamente la sintomatología del paciente.

Con la combinación de los resultados de los cuestionarios, la entrevistas personales, la experiencia del psiquiatra y el DSM III (Manual de Diagnóstico y Estadística de Trastornos Mentales), diagnostica si el paciente es depresivo, no depresivo o es depresivo como segundo diagnóstico.

El doctor Chappa realiza periódicamente el análisis general anteriormente descripto para evaluar la evolución del paciente.

Como conclusión, la información con la que contamos de cada paciente es una serie de puntajes (resultado de cada cuestionario) que califica aspectos de una categoría dentro de un conjunto finito de posibilidades.

Además se cuenta con el diagnóstico final y dos datos como la edad y el sexo, que de acuerdo a la experiencia del psiquiatra son considerados relevantes en la determinación del diagnóstico.

7.2) ¿Por que elegimos ID3?

Del survey surge que las dos estrategias candidatas para solucionar el problema son: aprendizaje a partir de ejemplos y aprendizaje por analogía (razonamiento basado en casos). Descartamos aprendizaje a partir de observación y descubrimientos porque, si usamos los casos psiquiátricos como observaciones estaríamos desaprovechando una información valiosa como la clasificación conocida. También descartamos aprendizaje a partir de instrucción porque el objetivo es tener una herramienta automática y no un sistema con el que el psiquiatra deba interactuar permanentemente.

De las dos estrategias candidatas mencionadas, elegimos aprendizaje a partir de ejemplos los casos psiquiátricos son tomados como ejemplos y no es necesario contar con información adicional (heurísticas), como en el caso de aprendizaje por analogía, en la que es necesario una métrica de similitud y las pistas derivacionales de las circunstancias en las que se diagnostica a un paciente.

Una vez seleccionada la estrategia de aprendizaje, se consulta nuevamente al survey para encontrar un método dentro de la estrategia seleccionada, que solucione el problema psiquiátrico. Consideramos que ID3 es el método más apropiado porque satisface los requerimientos del dominio:

- El objetivo del problema es obtener una regla de clasificación automática, ella es representada en ID3 a través de un árbol de decisión.
- Se deben asignar los casos a tres posibles diagnósticos, por los tanto se necesita un método como ID3 que aprenda múltiples conceptos disyuntos.
- Se cuenta con un conjunto de casos con una clasificación conocida, donde cada caso puede ser descripto por una conjunto fijo de atributos que pueden tomar valores predefinidos, que es lo necesario para aplicar ID3.

7.3) Modificaciones al método básico ID3

La metodología general del método ID3 ya fue desarrollada en "aprendizaje a partir de ejemplos" del survey. Existen algunas características del método que deben ser consideradas en una implementación particular:

- Cálculo de la información ganada.
- Tratamiento de los atributos desconocidos.
- Técnica para podar el árbol.
- Windowing.

7.3.1) Cálculo de información ganada

El criterio original para elegir un atributo como raíz del árbol en cada paso utilizado por ID3 es el criterio llamado "gain". La teoría de la información que sostiene este criterio, puede ser dada por la sentencia "la información que contiene un mensaje depende de su probabilidad y puede ser medida en bits con el opuesto del logaritmo base dos de esa probabilidad".

La probabilidad del mensaje "un ejemplo de un conjunto de prueba S pertenece a la clase C" está dada por:

$$\frac{\text{frecuencia}(C, S)}{|S|}$$

donde frecuencia(C,S) es la cantidad de ejemplos en S que pertenecen a la clase C y |S| es la cantidad de ejemplos de S. Por lo tanto la información que dicho mensaje contiene es:

$$-\log_2\left(\frac{\text{frecuencia}(C, S)}{|S|}\right) \text{bits}.$$

La cantidad de información promedio necesaria para identificar la clase de un caso en training set está dada por:

$$\text{info}(S) = -\sum_{i=1}^{k} \frac{\text{frecuencia}(C_i, S)}{|S|} * \log_2\left(\frac{\text{frecuencia}(C_i, S)}{|S|}\right) \text{bits}.$$

donde k es la cantidad de clases.

Ahora, similarmente, se calcula la información necesaria para identificar la clase de un caso en training set, después que el fue dividido de acuerdo a los n valores de un atributo X. Esto se define por la suma pesada de los n subconjuntos:

$$info_x(S) = -\sum_{i=1}^{n} \frac{|S_i|}{|S|} * info(S_i)$$

donde $|S_i|$ representa la cantidad de ejemplos de S que contiene el i-ésimo valor de X.

Por lo tanto

$$gain(X) = info(S) - info_X(S)$$

mide la información ganada seleccionando al atributo X como raíz del árbol. Luego se selecciona el atributo X que maximice gain(X).

Este cálculo de la información ganada tiene la desventaja que prefiere atributos con muchos valores posibles. Por ejemplo, en el caso que un atributo x_1 tenga un valor distinto para cada ejemplo, $info_{X1}(S) = 0$, por lo tanto x_1 es el atributo seleccionado, aunque dicha división no es útil.

Para evitar esto, se normaliza el cálculo de la información ganada con el cálculo de la información contenida en el mensaje que indica, en vez de la clase a la que pertenece el caso, el subconjunto al que pertenece, llamado "split info(X)":

$$split\ info(X) = -\sum_{i=1}^{n} \frac{|S_i|}{|S|} * \log_2\left(\frac{|S_i|}{|S|}\right)$$

Esta normalización lleva a un nuevo criterio llamado "gain ratio(X)"

$$gain\ ratio(X) = gain(X) / split\ info(X)$$

Si la división del conjunto de prueba es casi trivial (todos o casi todos los subconjuntos resultantes contienen sólo un ejemplo), split info es muy pequeño, por lo tanto gain ratio es muy grande. Para evitar seleccionar atributos que lleven a divisiones triviales, se considera sólo aquellos atributos cuya información ganada es mayor que el promedio de las informaciones ganadas de todos los atributos.

7.3.2) Tratamiento de los atributos desconocidos

El algoritmo original supone que todos los ejemplos tienen sus atributos con valores conocidos. Pero en situaciones reales, como en el dominio considerado, esto no siempre sucede. Esto lleva a la necesidad de modificar los ejemplos o modificar el algoritmo.

a) Modificación de los ejemplos: el objetivo es asignarle un valor al atributo cuyo valor es desconocido. Existen varios criterios:

1) Considerar "desconocido" como un posible valor de atributo.

2) Asignarle el valor más probable considerando los ejemplos con valor conocido en el training set.

3) Aplicar el método ID3 para averiguar los valores desconocidos de un atributo. Los ejemplos son formados a partir de los que tengan un valor conocido de este atributo, donde sus atributos son los demás atributos más la clase y su clase es el atributos a averiguar.

b) Modificación del algoritmo: tres puntos del algoritmo deben ser modificados:

1) Modificar el cálculo de la información ganada: para los ejemplos cuyos valores del atributo en cuestión son conocidos, info(S) y $info_X(S)$ son calculados de la misma manera. En cambio los ejemplos que tienen valor desconocido no brindan ninguna información. Por lo tanto, el criterio "gain" es definido ahora como:

$$gain(X) = PC(X) * (info(S) - info_X(S)) + PD(X) * 0$$

donde PC(X) es la probabilidad de que X sea conocido y PD(X) es la probabilidad de que X sea desconocido

Para el criterio gain ratio, el cálculo de split info es modificado considerando un subconjunto más para los desconocidos:

$$\text{split info}(X) = -\sum_{i=1}^{n} \frac{|S_i|}{|S|} * \log_2\left(\frac{|S_i|}{|S|}\right) + \frac{|D|}{|S|} * \log_2\left(\frac{|D|}{|S|}\right)$$

donde $|D|$ es la cantidad de desconocidos.

2) Modificar el particionamiento del training set: cuando se particiona el training set en base a un atributo X se debe decidir "que hacer" con los ejemplos cuyo valor de X es desconocidos. Cuando un ejemplo tiene un valor conocido x_i de atributos X, la probabilidad que este ejemplo pertenezca al i-ésimo subconjunto es 1, pero cuando este valor es desconocido dicha probabilidad debe ser calculada. Para esto se asocia a cada ejemplo de cada subconjunto un "peso" que representa la probabilidad que dicho ejemplo pertenezca a dicho subconjunto. Un ejemplo cuyo valor del atributo X es desconocido es asignado a todo subconjunto con su correspondiente peso, donde el peso de los ejemplos para el i-ésimo subconjunto es calculado como:

peso anterior * probab. que $\in$ al i-ésimo subconjunto

donde esta probabilidad es estimada como:

$$\frac{\text{suma de los pesos de los ejemplos con valor conocido i}}{\text{suma de los pesos de los ejemplos con valor conocido}}$$

Cada subconjunto S_i es ahora una colección de casos fraccionales, de modo que $|S_i|$ debe ser reinterpretado como la suma de los pesos de los ejemplos de $|S_i|$.

3) Clasificación de un ejemplo con valores de atributos desconocidos: cuando se debe testear sobre un atributos cuyo valor en el ejemplo es desconocido se deben explorar todos los "caminos" posibles y luego combinar las clasificaciones resultantes. De esta combinación resulta una distribución de clases; la clase con probabilidad más alta es la clase elegida.

7.3.3) Técnica para podar el árbol

El algoritmo original crea una hoja del árbol a partir de un nodo cuando todos los ejemplos en dicho nodo son de la misma clase. Cuando se trabaja con muchos ejemplos, esto puede

llevar a árboles de decisión demasiado profundos. Para evitar esto se puede aplicar la siguiente técnica de podado al costo de perder precisión: se fija un número mínimo de ejemplos para que un nodo sea subdividido si es que ellos son de distinta clase. Cuando tiene menos que esa cantidad de ejemplos, este nodo será una hoja del árbol donde la clase asociada a esta hoja puede ser:

1) La clase mayoritaria de los ejemplos de la hoja o
2) En vez de una clase única, asociar a la hoja una distribución de clases de acuerdo a la distribución de la clase de los ejemplos. Si cuando se clasifica un ejemplo se llega a esta hoja, se le asigna a dicho ejemplo la clase con mayor probabilidad (la diferencia entre estas dos propuestas sólo se refleja cuando se clasifican ejemplos con valores de atributos desconocidos).

Si algunos de los ejemplos no mantiene la relación de todo el training set entre los atributos y la clase (está fuera del patrón que se quiere representar), con el algoritmo original estos son representados en el árbol, lo que lleva a generar subestructuras adicionales que si no fuera por estos ejemplos no existirían. La técnica anterior también permite evitar que estos ejemplos (posiblemente incorrectos) influyan sobre el árbol de decisión final.

7.3.4) Windowing

En el prototipo desarrollado no se aplica la estrategia del uso de "window", porque como no se cuenta con muchos ejemplos no hay una mejora de tiempo computacional significativa.

7.4) Resultados

El programa ¨"id3" desarrolla al prototipo mencionado anteriormente. Dicho programa invoca a cuatro versiones que implementan algunas de las modificaciones al método básico id3 descriptas en el punto 7.4). Cada versión realiza 1000 corridas para que los resultados obtenidos tengan validez y no sean producto de coincidencias. Cada corrida crea un árbol de decisión, dejando de lado un porcentaje de los ejemplos, los cuales son utilizados para testear

dicho árbol de decisión. Ellos son clasificados por el árbol y el resultado obtenido es comparado con la clasificación conocida del ejemplo.

Las siguientes características son compartidas por las cuatro versiones:

a) Se utilizaron 60 ejemplos (casos psiquiátricos, ver Apéndice A).
b) Se utilizaron, como atributos, 8 de los cuestionarios que definen los aspectos psiquiátricos (ver Apéndice B).
c) La información ganada fue calculada con el criterio "gain ratio".
d) El 90% de los ejemplos fueron utilizados para crear el árbol de decisión y el 10% restante fueron utilizados para testearlo.

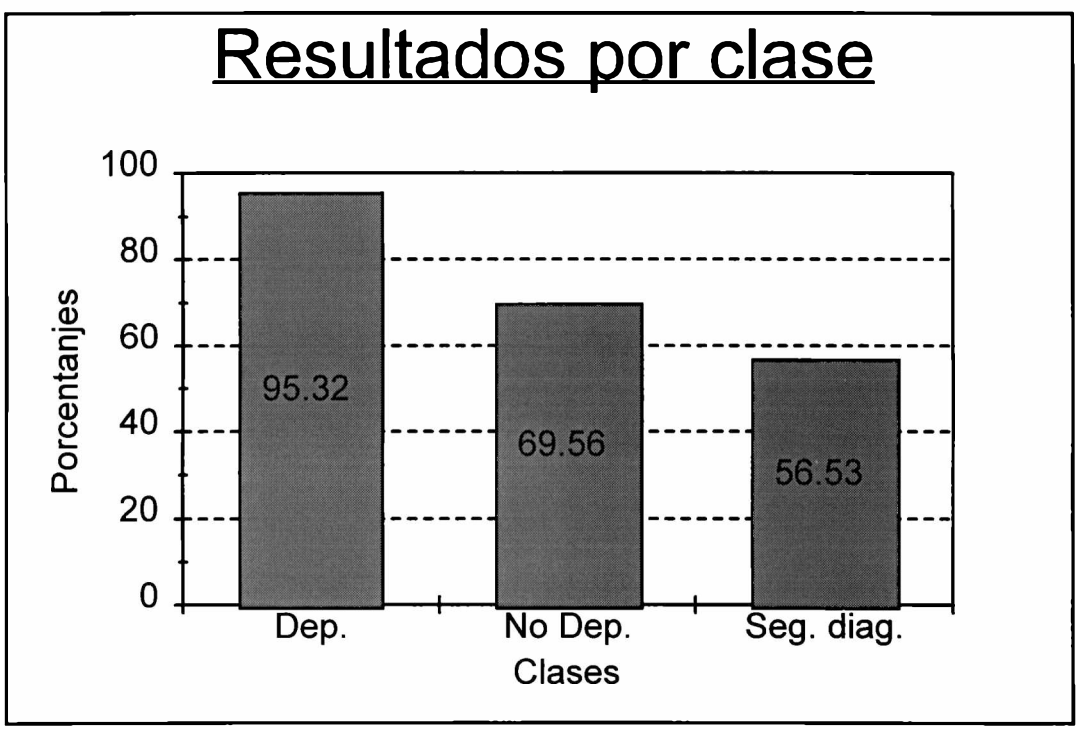
A continuación se detallan las características distintivas de cada versión y los resultados obtenidos.

Versión 1

- Los atributos desconocidos (cuestionarios que no fueron contestado o que no fueron registrados) son tratados modificando el algoritmo (opción b) del punto 3.2)), solo que se utilizaron como ejemplos de test sólo ejemplos con todos los atributos conocidos.

- No se aplicó ninguna técnica de podado.

Con estos datos se clasificó correctamente el **87.00%** de ejemplos de test. El siguiente gráfico muestra como se distribuyen las clasificaciones en cada clase.



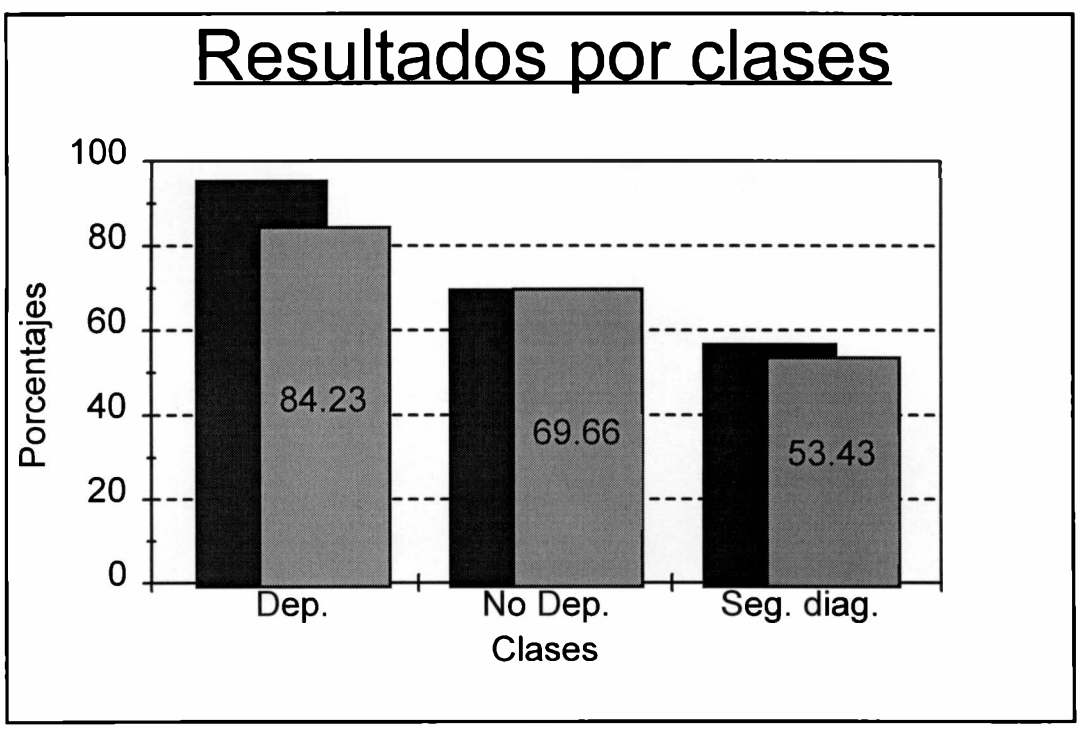
Las versiones siguientes son comparadas con los resultados de la versión 1 (Version con la que se obtuvieron mejores resultados)

Versión 2

- Los atributos desconocidos (cuestionarios que no fueron contestado o que no fueron registrados) son tratados modificando el algoritmo (opción b) del punto 3.2)), solo que se utilizaron como ejemplos de test sólo ejemplos con todos los atributos conocidos.

- Se aplica la técnica de podado donde el número mínimo de ejemplos para que un nodo sea subdividido si es que ellos son de distinta clase es 3, donde la clase asociada a la hoja es la clase mayoritaria de los ejemplos en ella.

Con esta modificación se clasificó correctamente el **78,55%** de ejemplos de test. Pero, mientras el árbol de decisión del experimento original tiene **23** nodos de test de promedio, el árbol de decisión de este experimento tiene **14** nodos de test de promedio. El siguiente gráfico muestra como se distribuyen las clasificaciones en cada clase (en comparación con el resultado original).

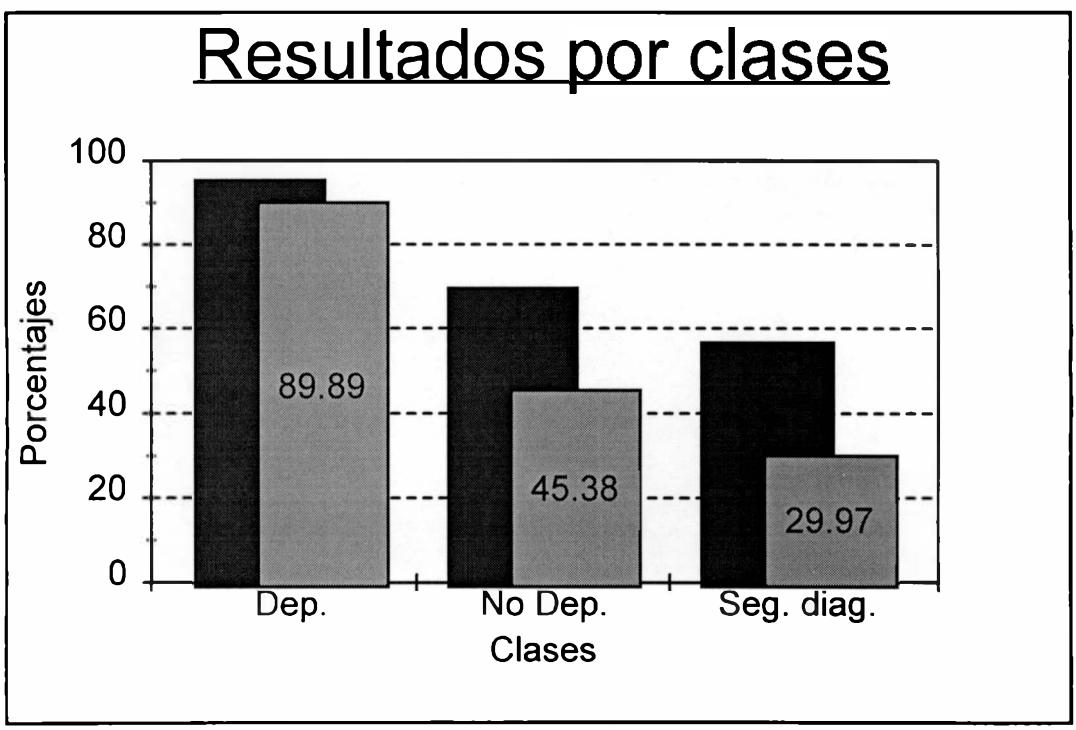


Versión 3

- Los atributos desconocidos (cuestionarios que no fueron contestado o que no fueron registrados) son tratados modificando el algoritmo (opción b) del punto 3.2)), solo que se utilizaron como ejemplos de test sólo ejemplos con todos los atributos conocidos.

- Se aplica la técnica de podado donde el número mínimo de ejemplos para que un nodo sea subdividido si es que ellos son de distinta clase es 3, donde la clase asociada a la hoja una distribución de clases de acuerdo a la distribución de la clase de los ejemplos.

Con esta modificación se clasificó correctamente el **74.17%** de ejemplos de test. Pero, mientras el árbol de decisión del experimento original tiene **23** nodos de test de promedio, el árbol de decisión de este experimento tiene **13** nodos de test de promedio. El siguiente gráfico muestra como se distribuyen las clasificaciones en cada clase (en comparación con el resultado original).

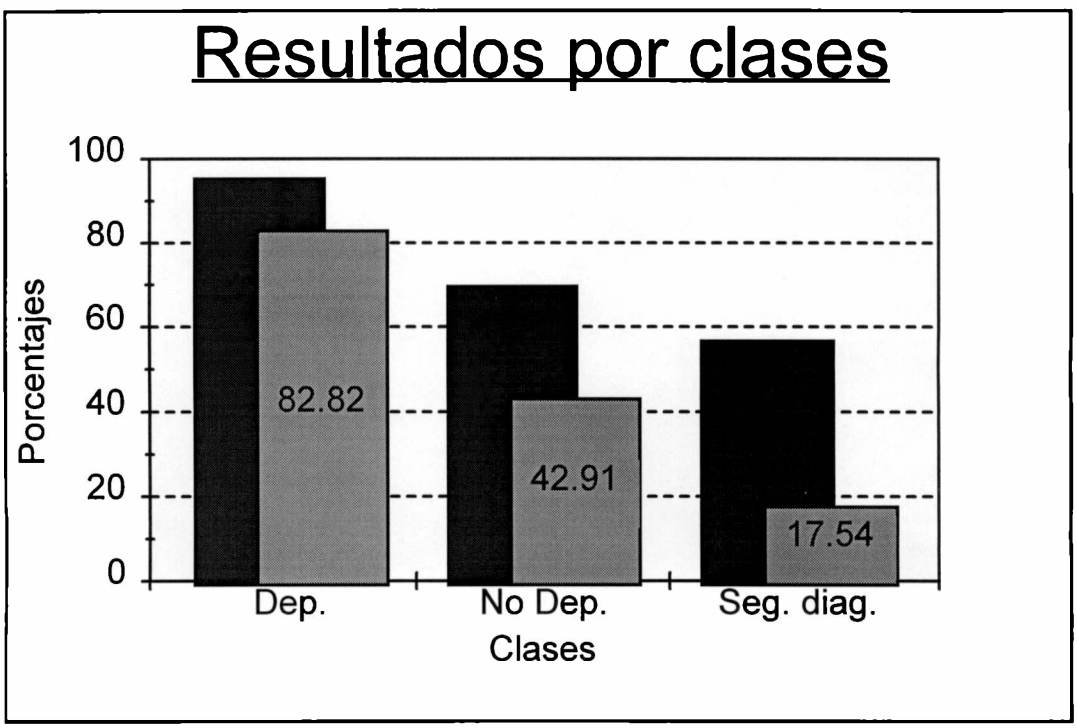


Versión 4

- Los atributos desconocidos (cuestionarios que no fueron contestado o que no fueron registrados) son tratados modificando los ejemplos, considerando al desconocido como un posible valor (opción a)1) del punto 3.2)).

- No se aplicó ninguna técnica de podado.

Con esta modificación se clasificó correctamente el **67,55%** de ejemplos de test. El siguiente gráfico muestra como se distribuyen las clasificaciones en cada clase (en comparación con el resultado original).



Comentario final y evolución futura

Una vez concluido el trabajo creemos que las expectativas iniciales fueron satisfechas: se desarrolló un survey de aprendizaje computacional y él fue útil para encontrar una estrategia y un método que resolviera satisfactoriamente el problema de clasificar pacientes psiquiátricos.

El trabajo podría ser utilizado en distintas áreas para facilitar la incorporación de estas técnicas en sistemas de software donde el sistema final trate con un dominio complejo que se adapte más a métodos adaptativos que a enfoques rígidos de programación algorítmica. También podría aplicarse a sistemas que interactúan con individuos con fines de capacitación y donde la tipificación de actividades solo tiene sentido a partir de un enfoque que incorpore facilidades basada en el análisis de casos.

Una posible evolución del trabajo sería la automatización del survey: a partir de la especificación de un problema real obtener la/s métodologías más apropiadas para resolverlo. Está extensión se vería beneficiada por contar con nuestro análisis como punto de partida y poder concentrarse en focalizar el trabajo en la construcción de un herramienta de alto nivel que asesore al diseñador o/y implementador en la selección de la técnica más conveniente para cada fin.

Apéndice A

La siguiente tabla muestra los puntajes de los pacientes en cada cuestionario:

Ind	Clase	Sexo	Edad	Cu1	Cu2	Cu3	Cu4	Cu5	Cu6	Cu7	Cu8
1	2	F	16	20	28	20	24	15	20	6	51
2	0	F	18	8	11	2	22	17	21	1	33
3	0	M	25	7	18	5	23	15	16	3	18
4	0	M	32	33	27	-	29	35	23	8	44
5	0	M	24	6	13	1	-	-	-	-	-
6	0	M	34	17	21	17	24	17	31	6	46
7	2	F	22	31	36	35	30	16	20	7	109
8	0	F	18	8	9	13	20	13	34	4	59
9	0	F	28	23	34	31	46	20	17	10	65
10	0	F	25	19	27	8	38	4	8	2	104
11	0	F	35	8	13	5	27	21	20	6	33
12	1	F	25	21	27	16	29	25	40	6	51
13	2	F	66	18	-	8	13	4	15	8	24
14	0	M	34	26	39	24	36	14	19	4	43
15	0	F	20	13	18	11	28	18	28	8	62
16	0	F	41	34	35	27	8	36	8	5	85
17	0	F	28	31	14	1	-	-	-	-	131
18	0	F	30	16	14	10	34	16	41	7	35
19	0	M	-	17	24	6	31	23	26	11	20

Ind	Clase	Sexo	Edad	Cu1	Cu2	Cu3	Cu4	Cu5	Cu6	Cu7	Cu8
20	0	M	65	12	17	3	28	17	23	7	18
21	0	M	40	3	11	0	23	14	30	0	14
22	0	F	37	41	45	36	36	15	22	12	93
23	0	F	44	13	-	11	25	10	6	8	38
24	0	F	38	19	36	27	26	30	17	10	32
25	0	M	32	16	20	6	34	17	23	10	24
26	0	F	20	18	21	14	45	37	24	8	51
27	1	F	42	34	25	27	10	10	5	7	45
28	1	F	35	8	23	17	-	-	-	-	20
29	1	F	35	34	38	28	28	17	13	11	27
30	2	M	-	24	31	16	32	25	25	3	66
31	1	M	35	37	45	38	33	23	21	11	95
32	1	F	36	20	34	16	26	2	4	4	27
33	2	F	26	29	41	34	23	12	21	7	49
34	1	M	18	27	-	15	8	2	13	4	29
35	0	M	41	19	-	-	-	-	-	-	35
36	0	M	31	4	5	5	-	-	-	-	30
37	0	F	56	17	16	11	25	17	23	4	15
38	0	F	-	20	26	29	45	45	28	8	49
39	0	M	47	5	9	2	-	-	-	-	8
40	1	F	18	28	-	44	30	27	37	8	100
41	0	F	24	28	28	21	31	18	18	5	82
42	0	F	36	7	13	5	-	-	-	-	33
43	1	F	41	15	-	16	41	19	21	8	23

Ind	Clase	Sexo	Edad	Cu1	Cu2	Cu3	Cu4	Cu5	Cu6	Cu7	Cu8
44	2	F	-	32	38	41	37	21	23	7	79
45	0	M	28	22	-	23	48	32	43	6	106
46	0	F	43	32	37	23	35	24	22	11	94
47	0	F	49	30	-	22	17	0	16	4	58
48	0	F	51	6	-	6	17	7	14	7	61
49	1	F	68	30	-	33	34	25	25	9	79
50	0	M	27	15	19	10	-	-	-	-	53
51	0	F	38	5	3	3	19	15	31	5	24
52	0	M	54	24	-	-	3	14	13	8	27
53	0	M	32	13	27	15	-	-	-	-	13
54	0	F	25	21	24	-	33	21	31	8	56
55	1	F	32	21	24	19	26	19	28	5	24
56	0	M	-	-	-	14	27	24	20	7	20
57	0	M	48	3	5	3	15	18	25	0	28
58	0	F	28	39	37	46	47	28	27	12	86
59	1	F	65	23	-	23	22	1	2	3	25
60	1	F	37	-	-	23	27	19	28	9	67

Apéndice B

Los cuestionarios utilizados en los experimentos son:

1) C.R.S.: mide sentimientos y pensamientos recientes y su clasificación es:

Normal (clase 0)	→ puntaje de 0 a 10.
Con alteraciones (clase 1)	→ puntaje de 11 a 52.

2) E.D.A.M.: mide sentimientos y pensamientos recientes y su clasificación es:

Normal (clase 0)	→ puntaje de 0 a 14.
Con alteraciones (clase 1)	→ puntaje de 15 a 53.

3) Beck: evalúa el aspecto depresivo y su clasificación es:

Sin depresión o mínima (clase 0)	→ puntaje de 0 a 9.
Depresión Borderline (clase 1)	→ puntaje de 10 a 14.
Depresión leve (clase 2)	→ puntaje de 15 a 20.
Depresión moderada (clase 3)	→ puntaje de 21 a 30.
Depresión severa (clase 4)	→ puntaje de 31 a 40.
Depresión muy severa (clase 5)	→ puntaje de 41 a 63.

4) Escala de atribución psicológica: su clasificación es:

Normal (clase 0)	→ puntaje de 0 a 27.
Con alteraciones (clase 1)	→ puntaje de 28 a 52.

5) Escala de atribución somática: su clasificación es:

Normal (clase 0)	→ puntaje de 0 a 19.
con alteraciones (clase 1)	→ puntaje de 20 a 52.

6) Escala de atribución normal: su clasificación es:

Normal (clase 0)	→ puntaje de 0 a 34.
Con alteraciones (clase 1)	→ puntaje de 35 a 52.

7) Escala de atribución sintomática: su clasificación es:

Normal (clase 0) → puntaje de 0 a 7.

Con alteraciones (clase 1) → puntaje de 8 a 13.

8) Inventario de Padua: mide evitación y agarofovia y su clasificación es:

Normal (clase 0) → puntaje de 0 a 53.

Con alteraciones(clase 1) → puntaje de 54 a 240.

A continuación se adjunta cada uno de estos cuestionarios.

ESCALA C.R.S.

Instrucciones: *Estas son algunas frases que describen el modo como pudo haberse sentido durante la última semana. Léalas atentamente y responda en el casillero correspondiente (SI-NO) de acuerdo a cómo se ha sentido. No omita ninguna respuesta. Cuando tenga dudas, responda con la alternativa que más se aproxime al modo como se ha sentido.*

	SI	NO
1) Me siento con tantas energías como siempre	☐	☐
2) Estoy adelgazando	☐	☐
3) He abandonado muchas actividades y perdido interés en muchas cosas	☐	☐
4) Desde que me siento enfermo he perdido por completo interés en el sexo	☐	☐
5) Estoy muy preocupado por el funcionamiento de mi organismo	☐	☐
6) Se debe notar que estoy perturbado y agitado	☐	☐
7) Aún soy capaz de seguir trabajando	☐	☐
8) Me concentro fácilmente al leer el diario	☐	☐
9) Me lleva más de media hora dormirme	☐	☐
10) Me siento inquieto y agitado	☐	☐
11) Me despierto más temprano que lo necesario	☐	☐
12) La mejor solución es que me muera	☐	☐
13) Siento mareos y como si fuera a desmayarme	☐	☐
14) Estoy siendo castigado por algo malo que he hecho	☐	☐
15) Mi interés por el sexo es igual que antes de sentirme mal	☐	☐
16) Me siento mal, desdichado o con ganas de llorar	☐	☐
17) A veces deseo estar muerto	☐	☐
18) Tengo problemas digestivos	☐	☐
19) Me despierto en la mitad de la noche	☐	☐
20) Me siento inútil y me avergüenzo	☐	☐
21) Me siento tan decaído que necesito ayuda para bañarme y cambiarme	☐	☐
22) Me cuesta más que antes dormirme	☐	☐
23) Siento temor pero no sé el motivo	☐	☐
24) Me acosan remordimientos	☐	☐
25) Me agrada y produce satisfacción todo lo que hago	☐	☐
26) Todo lo que necesito es un buen descanso para sentirme bien otra vez	☐	☐
27) Mi sueño es intranquilo y no descanso	☐	☐
28) Siento la mente tan clara y rápida como siempre	☐	☐
29) Pienso que la vida vale la pena	☐	☐
30) Hablo sin entusiasmo y mi voz suena apagada	☐	☐
31) Me siento irritable o nervioso	☐	☐
32) Tengo buen ánimo	☐	☐
33) A veces siento que mi corazón se acelera	☐	☐
34) Creo que no tengo solución	☐	☐
35) Me despierto más temprano que de costumbre en las mañanas	☐	☐
36) Disfruto de las comidas como siempre	☐	☐
37) Me paso el día dando vueltas	☐	☐
38) Me siento casi aterrorizado	☐	☐
39) Mi cuerpo funciona mal y tengo algo malo adentro	☐	☐
40) Me enfermo por los malos tiempos que corren	☐	☐
41) Mis manos tiemblan tanto que cualquiera se da cuenta	☐	☐
42) Me gusta salir y estar con gente	☐	☐
43) Me parece que exteriormente estoy calmo	☐	☐
44) Pienso que soy tan bueno como cualquiera	☐	☐
45) Mi problema es una enfermedad seria que tengo adentro	☐	☐
46) He estado pensando en matarme	☐	☐
47) Apenas puedo hacer algo últimamente	☐	☐
48) Mi futuro me parece completamente miserable	☐	☐
49) Me preocupo mucho por mis malestares físicos	☐	☐
50) Tengo que hacer un gran esfuerzo para comer algo	☐	☐
51) Me siento casi siempre cansado	☐	☐
52) He perdido mucho peso	☐	☐

C.E.T.E.M.
Dr. Herbert J. Chappa
Director

E.D.A.M.

Nombre y Apellido.....
P. Nº.....Edad.....Estado Civil.....
Sexo.....N. Ed.....Diagnóstico.....
Fármaco.....Fecha...../.....

PUNTAJE:

INSTRUCCIONES

A continuación encontrará una serie de frases que describen distintas formas en las que ud. pudo haber actuado, sentido o pensado durante la última semana. Léalas y vea si ellas coinciden con lo que Ud. pudo haber hecho, sentido o pensado durante la última semana.

Si Ud. considera que en efecto corresponde a lo que le ha pasado, ponga una cruz en el casillero correspondiente a SI; en caso contrario colóquela en el casillero correspondiente a NO.

Trabaje rápido y no demore mucho tiempo en cada frase. Salvo que no comprenda exactamente el significado deje los comentarios para cuando finalice el test. Toda la prueba no debe llevarle más de unos pocos minutos. Tenga cuidado de no dejar de contestar ninguna frase.

Ahora de vuelta la página y comience. Trabaje con rapidez y recuerde que debe contestar todas las frases. No hay respuestas correctas o equivocadas. Se trata simplemente de ver como Ud. se ha sentido durante la última semana.

Durante la última semana:

	SI	NO
1) Me he cansado rápidamente.....		
2) Me sentí mal del estómago.....		
3) Me sentí tan nervioso como cualquier persona.....		
4) Tuve muy pocos dolores de cabeza.....		
5) Trabajé bajo gran tensión.....		
6) Tuve dificultades para concentrarme.....		
7) Prácticamente nunca me ruboricé.....		
8) Sentí mis manos y pies lo suficientemente calientes...		
9) Noté que mis manos temblaban cuando trataba de hacer algo.....		
10) Me ruboricé con la misma frecuencia que los demás....		
11) Tuve diarrea por lo menos una vez.....		
12) Me sentí preocupada por posibles problemas.....		
13) Con frecuencia tuve temor de ponerme colorado.....		
14) Tuve pesadillas.....		
15) Me preocupé mucho por el dinero y mis ocupaciones....		
16) Me sentí muy nervioso.....		

Durante la última semana:

	SI	NO
17) Sentí rara vez palpitaciones o dificultades para respirar.....		
18) Sentí hambre casi permanentemente.....		
19) Pasé varios días sin mover el vientre.....		
20) Me sentí casi siempre ansioso por algo o por alguien.		
21) Cuando me sentí perturbado transpiré en una forma muy molesta.....		
22) A veces perdí el sueño por las preocupaciones.....		
23) Me sentí molesto y perturbado frente a otras personas		
24) Soñé cosas que no me gustaría contar a los demás.....		
25) Me encontré, con frecuencia, preocupado en exceso por algo.....		
26) He deseado poder ser tan feliz como los demás.....		
27) Por lo general me sentí tranquilo y no me puse nervioso fácilmente.....		
28) Lloré fácilmente.....		
29) Mi sueño fue intranquilo y perturbado.....		
30) Tuve muchos problemas digestivos.....		
31) Cuando tuve que esperar me puse nervioso.....		
32) Pensé que afrontaba dificultades que no podía resolver.....		

Durante la última semana:

	SI	NO
33) Me sentí con tantos temores como siempre.....		
34) Me sentí herido en mis sentimientos con mayor facilidad que otras personas.....		
35) A veces me sentí tan nervioso que me resultó difícil dormirme.....		
36) Por lo general me sentí feliz.....		
37) He estado preocupado por cosas que realmente no tenían importancia.....		
38) A veces estuve tan inquieto que no podía quedarme sentado en una silla mucho tiempo.....		
39) Estuve tan temeroso como siempre.....		
40) Me resultó difícil concentrarme en una tarea o trabajo.....		
41) En general me tuve confianza.....		
42) Mis relaciones sexuales fueron satisfactorias.....		
43) A veces me sentí inútil.....		
44) Me he sentido fácilmente inhibido.....		
45) He tomado las cosas a la tremenda.....		
46) Transpiro con facilidad aún si el día estaba fresco.		
47) He tenido muy poco deseo sexual.....		
48) He pensado que vivir era un esfuerzo para mí.....		

Durante la última semana:

	SI	NO
49) En general he tenido mucha confianza en mi mismo...		
50) Pensé que realmente no soy bueno.....		
51) Pensé que me estaba viniendo abajo.....		
52) Evité enfrentar dificultades o tomar decisiones importantes.....		
53) Perdí interés por el sexo.....		

CETEM

INVENTARIO DE LA DEPRESION DE BECK

Nombre y Apellido..... Fecha.....

Responda cada una de las preguntas de este cuestionario, en el que aparecen varios grupos de afirmaciones, que mejor describan cómo se ha sentido esta semana, incluyendo el día de hoy. De haber más de una afirmación, señálela también:

1. 0 No me siento triste.
   1 Me siento triste.
   2 Me siento triste constantemente, y no puedo evitarlo.
   3 Me siento tan triste o tan desgraciado que no puedo tolerarlo.

2. 0 No me siento particularmente desanimado con respecto al futuro.
   1 Me siento desanimado con respecto al futuro.
   2 Siento que no tengo nada por qué luchar.
   3 El futuro es desalentador y las cosas no van a mejorar.

3. 0 No me siento fracasado.
   1 He fracasado más que la mayoría de las personas.
   2 Cuando miro mi pasado, sólo veo un fracaso tras otro.
   3 Soy un fracaso total como persona.

4. 0 Las cosas me satisfacen tanto como antes.
   1 No disfruto de las cosas tanto como antes.
   2 No obtengo satisfacción real de las cosas.
   3 Estoy insatisfecho o aburrido con respecto a todo.

5. 0 No me siento particularmente culpable.
   1 Me siento culpable buena parte del tiempo.
   2 Me siento culpable la mayor parte del tiempo.
   3 Me siento culpable todo el tiempo.

6. 0 No siento que me estén castigando.
   1 Siento que podría ser castigado.
   2 Tengo la certeza de que seré castigado.
   3 Siento que me están castigando.

7. 0 No me siento decepcionado de mí mismo.
   1 Estoy decepcionado de mí mismo.
   2 Estoy a disgusto conmigo mismo.
   3 Me detesto.

8. 0 No siento que sea peor que otras personas.
   1 Me critico a mí mismo por mi debilidad y mis errores.
   2 Me culpo todo el tiempo por mis faltas.
   3 Me culpo por todo lo malo que sucede.

9. 0 No tengo ningún pensamiento suicida.
   1 Tengo pensamientos suicidas, pero no los llevaría a cabo.
   2 Desearía matarme.
   3 Me suicidaría si tuviera la oportunidad.

10. 0 No lloro más de lo habitual.
    1 Ahora lloro más que antes.
    2 Lloro todo el tiempo.
    3 Solía poder llorar, y ahora no puedo aunque quiera.

C E T E M
Institución de Psicoterapias Integrativas

Nombre y Apellido.....
Fecha...../...../.....
Edad..... años.. Sexo M ☐ F ☐ Condición Basal ☐ Sem ☐

ESCALA DE ATRIBUCIÓN PSICOLÓGICA

A continuación encontrará una lista de malestares que Ud. puede o no haber experimentado. Para cada uno, por favor haga un círculo en la letra correspondiente a la que corresponda a su respuesta más probable para cada caso. Verifique cada alternativa en cada una de la preguntas. También especifique si ese malestar lo ha tenido (-SI) o no (-NO) en los últimos tres meses. No deje preguntas sin contestar.

A	B	C	D
Para nada	Un poco	Bastante	Mucho

1. Si tuviera un dolor de cabeza prolongado, pensaría que estoy emocionalmente perturbado	A B C D
2. Si transpirara mucho, lo más probable es que piense que estoy muy nervioso o muy ansioso	A B C D
3. Si me sintiera de golpe mareado, pensaría que estoy muy estresado	A B C D
4. Si notara mi boca muy seca, pensaría que estoy ansioso o asustado por algo	A B C D
5. Si sintiera latir fuerte mi corazón, pensaría que me he puesto muy nervioso o sentí miedo	A B C D
6. Si me sintiera cansado, pensaría que estoy muy abatido o agotado	A B C D
7. Si notara que mis manos tiemblan, pensaría que estoy muy nervioso	A B C D
8. Si tuviera problemas para dormir, pensaría que estoy muy preocupado o nervioso por algo	A B C D
9. Si me sintiera mal del estómago, pensaría que es porque las preocupaciones me enfermaron	A B C D
10. Si perdiera el apetito pensaría que lo más probable sería que estoy tan preocupado que no tengo ganas de comer	A B C D
11. Si tuviera dificultades para respirar, pensaría que estoy ansioso o nervioso	A B C D
12. Si sintiera entumecimientos u hormigueos en las manos o en los pies, pensaría que estoy estresado	A B C D
13. Si estuviera constipado, pensaría que la nerviosidad me afecta el intestino	A B C D

C E T E M
Institución de Psicoterapias Integrativas

Nombre y Apellido.....
Fecha...../...../.....
Edad..... años.. Sexo M ☐ F ☐ Condición Basal ☐ Sem ☐

ESCALA DE ATRIBUCIÓN SOMÁTICA

A continuación encontrará una lista de malestares que Ud. puede o no haber experimentado. Para cada uno, por favor haga un círculo en la letra correspondiente a la que corresponda a su respuesta más probable para cada caso. Verifique cada alternativa en cada una de la preguntas. También especifique si ese malestar lo ha tenido (-SI) o no (-NO) en los últimos tres meses. No deje preguntas sin contestar.

A	B	C	D
Para nada	Un poco	Bastante	Mucho

1. Si tuviera un dolor de cabeza prolongado, pensaría que tengo algo en los músculos, los nervios o el cerebro	A B C D
2. Si transpirara mucho, lo más probable es que piense que debo tener fiebre o laguna infección	A B C D
3. Si me sintiera de golpe mareado, pensaría que me falla el corazón o que tengo presión alta	A B C D
4. Si notara mi boca muy seca, pensaría que tengo alguna enfermedad en mis glándulas salivares	A B C D
5. Si sintiera latir fuerte mi corazón, pensaría que debo tener una enfermedad del corazón	A B C D
6. Si me sintiera cansado, pensaría que puedo estar anémico o tener alguna enfermedad	A B C D
7. Si notara que mis manos tiemblan, pensaría que tengo alguna enfermedad neurológica	A B C D
8. Si tuviera problemas para dormir, pensaría que algún malestar o dolor me mantiene despierto	A B C D
9. Si me sintiera mal del estómago, pensaría que tengo gases o gastritis	A B C D
10. Si perdiera el apetito pensaría que lo más probable sería que tengo problemas gástricos o intestinales	A B C D
11. Si tuviera dificultades para respirar, pensaría que tengo un problema cardíaco o en los bronquios	A B C D
12. Si sintiera entumecimientos u hormigueos en las manos o en los pies pensaría que tengo problemas de circulación o en los nervios	A B C D
13. Si estuviera constipado, pensaría que tengo malo en mi intestino	A B C D

C E T E M
Institución de Psicoterapias Integrativas

Nombre y Apellido.....
Fecha...../...../.....
Edad..... años.. Sexo M ☐ F ☐ Condición Basal ☐ Sem ☐

ESCALA DE ATRIBUCIÓN NORMAL

A continuación encontrará una lista de malestares que Ud. puede o no haber experimentado. Para cada uno, por favor haga un círculo en la letra correspondiente a la que corresponda a su respuesta más probable para cada caso. Verifique cada alternativa en cada una de la preguntas. También especifique si ese malestar lo ha tenido (-SI) o no (-NO) en los últimos tres meses. No deje preguntas sin contestar.

A	B	C	D
Para nada	Un poco	Bastante	Mucho

1. Si tuviera un dolor de cabeza prolongado, pensaría que me afectó el ruido, las luces o alguna otra cosa A B C D
2. Si transpirara mucho, lo más probable es que piense que el lugar es muy caluroso, o estoy muy abrigado, o que he estado haciendo un trabajo intenso A B C D
3. Si me sintiera de golpe mareado, pensaría que no he comido lo suficiente o que me he levantado de golpe A B C D
4. Si notara mi boca muy seca, pensaría que necesito tomar líquidos A B C D
5. Si sintiera latir fuerte mi corazón, pensaría que he hecho mucho esfuerzo o tomado mucho café o mate A B C D
6. Si me sintiera cansado, pensaría que no hago ejercicios o que me he sobre-exigido A B C D
7. Si notara que mis manos tiemblan, pensaría que tengo cansado los brazos A B C D
8. Si tuviera problemas para dormir, pensaría que no estoy cansado o que tomé mucho café A B C D
9. Si me sintiera mal del estómago, pensaría que es porque he comido algo que me cayó mal A B C D
10. Si perdiera el apetito pensaría que lo más probable sería que he comido demasiado, o que necesito comer menos que antes A B C D
11. Si tuviera dificultades para respirar, pensaría que falta el aire en la habitación o que el aire está contaminado A B C D
12. Si sintiera entumecimientos u hormigueos en las manos o en los pies pensaría que se me durmió el pie o tengo frío A B C D
13. Si estuviera constipado, pensaría que no como suficientes verduras o fibras A B C D

C E T E M
Institución de Psicoterapias Integrativas

Nombre y Apellido.....
Fecha...../...../.....
Edad..... años.. Sexo M ☐ F ☐ Condición Basal ☐ Sem ☐

ESCALA DE ATRIBUCIÓN SINTOMÁTICA

A continuación encontrará una lista de malestares que Ud. puede o no haber experimentado. Para cada uno, por favor haga un círculo en la letra correspondiente a la que corresponda a su respuesta más probable para cada caso. Verifique cada alternativa en cada una de la preguntas. También especifique si ese malestar lo ha tenido (-SI) o no (-NO) en los últimos tres meses. No deje preguntas sin contestar.

1. Ha tenido un dolor de cabeza persistente en los últimos tres meses?	S / N
2. Ha notado que transpiraba mucho en los últimos tres meses?	S / N
3. Se ha sentido mareado en los últimos tres meses?	S / N
4. Ha notado por momentos la boca muy seca en los últimos tres meses?	S / N
5. Tuvo palpitaciones en los últimos tres meses?	S / N
6. Se ha sentido cansado en los últimos tres meses?	S / N
7. Ha notado temblor en sus manos en los últimos tres meses?	S / N
8. Ha tenido problemas para dormir en los últimos tres meses?	S / N
9. Ha tenido malestares de estómago en los últimos tres meses?	S / N
10. Ha perdido el apetito en los últimos tres meses?	S / N
11. Ha sentido falta de aire en los últimos tres meses?	S / N
12. Ha sentido entumecimiento u hormigueos en los últimos tres meses?	S / N
13. Estuvo constipado o irregular del intestino en los últimos tres meses?	S / N

C E T E M

INVENTARIO PADUA

Nombre y Apellido.....
Fecha...../...../.....
Edad..... años.. Sexo M ☐ F ☐ Condición Basal ☐ Sem ☐

INSTRUCCIONES: Las frases siguientes hacen referencia a algunos pensamientos, ideas, acciones o temores que cualquiera puede tener a diario. Para cada frase elija la respuesta que mejor describe lo que a Ud. le sucede y el grado en que lo perturban tales acciones o pensamientos. Marque entonces la respuesta, ala izquierda de la frase, con los siguientes números:

0= Para nada
1= Un poco
2= Bastante
3= Mucho
4= Muchísimo

Siento mis manos sucias cuando toco dinero
Creo que el menor contacto con mis secreciones(transpiración, saliva, orina, etc) puede contaminar mis ropas
Me resulta dificil tocar algún objeto cuando ha sido tocado por extraños o por determinadas personas
Me cuesta tocar desperdicios o cosa sucias
Evito usar baños públicos por temor a contagiarme o contraer enfermedades
Evito usar teléfonos públicos por temor a contagiarme o enfermar
Me lavo las manos más veces y por más tiempo que lo necesario
A veces tengo que lavarme porque pienso que estoy sucio o contaminado
Si toco algo que pienso que está contaminado tengo que lavarme o limpiarme inmediatamente
Si un animal, me toca o me roza, me siento sucio y tengo que lavarme y cambiarme de ropa inmediatamente
Cuando tengo dudas o preocupaciones no puedo estar tranquilo hasta que lo hablo con alguien que me da seguridad
Tiendo a repetir las mismas cosas o las mismas frases cuando hablo
Tengo la costumbre de pedir que me repitan las cosas aunque haya entendido la primera vez
No puedo dejar de seguir un cierto orden cuando me visto, me desvisto o me baño
Antes de ir a dormir tengo que hacer ciertas cosas en cierto orden
Antes de meterme en la cama tengo que doblar la ropa de cierta manera
A veces tengo que repetir número mentalmente sin ninguna razón
Tengo que hacer las cosas varias veces para asegurarme de que estén bien hechas
Tiendo a controlar las cosas con más frecuencia que lo necesario
Controlo las llaves del gas o las luces varias veces cuando las cierro o apago
He tenido que volver a mi casa (u oficina) para asegurarme que las puertas o las ventanas (o luces, llaves, etc) quedaron cerradas
Leo y releo planillas, documentos, cheques, etc una y otra vez para estar seguro que están bien hechos
Tengo que controlar que los fósforos o colillas, quedaron bien apagados
Cuando manejo dinero lo cuento y recuento una y otra vez
Controlo las cartas varias veces antes de enviar al correo
Me resulta dificil tomar decisiones aún ante cosas simples
A veces no estoy seguro si hice algo que realmente se que he hecho
Al hablar, tengo la impresión de que jamás se explicarme con claridad, especialmente cuando se trata de algo importante
Después de hacer algo cuidadosamente, con la impresión de haberlo hecho mal o que no lo terminé
A menudo llego tarde porque me demoro haciendo ciertas cosas por más tiempo que el necesario
Me invento dudas y problemas con la mayor parte de las cosas que hago
Cuando empiezo a pensar en algo me obsesiono con ello
Me viene ideas malas contra mi voluntad y no las puedo evitar
Pienso malas palabras y no me las puedo sacar de la cabeza

Se me va la mente a otras cosas, y me cuesta prestar atención a lo que sucede a mi alrededor
Imagino consecuencias catastróficas por mis distracciones o por mi más pequeño error
Me preocupa mucho pensar que pude herir a alguien sin saberlo
Cuando me entero de algún desastre (catástrofe) pienso que pudo ser mi culpa
A veces me preocupo sin razón creyendo tener alguna enfermedad o que me han hecho un daño
A veces, empiezo a contar objetos sin ninguna razón
Creo que algunas veces tengo que memorizar números que no son importantes
Cuando leo me parece que he omitido algo importante y tengo que volver a leer dos o tres veces
Me preocupo por acordarme cosas sin importancia y hago un esfuerzo por no olvidarlas
Cuando me viene un pensamiento o una duda, tengo que dar vueltas y vueltas, y no puedo parar hasta analizarla desde todos los ángulos
En ciertas ocasiones tengo miedo de perder el control y cometer torpezas
Cuando miro desde un lugar alto, como una ventana, un balcón o un puente siento como el impulso de tirarme
Cuando se acerca un tren pienso si no podría tirarme bajo las ruedas
En ciertos momentos, tengo el impulso de quitarme la ropa en público
Cuando manejo siento el impulso de tirar el auto contra alguien o contra algo
Cuando veo armas de fuego me pongo nervioso y tengo ideas agresivas o de violencia
Me hace mal ver cuchillos, tijeras u otros objetos puntiagudos
A veces siento que algo me empuja a hacer cosas sin sentido, que no quiero hacer
Necesito a veces, romper cosas o dañarlas, sin ningún motivo
A veces tengo el impulso de hurtar algo de otro, aunque no lo necesite
Cuando voy al supermercado tengo la tentación de llevarme algo
Siento un impulso, a veces, como de dañar a un animal o a un niño indefenso
Tengo la necesidad, a veces, de caminar de cierta manera o hacer ciertos gestos
De vez en cuando, me descontrolo para comer, aún cuando me siento mal después
Cuando me entero de algún crimen o de un suicidio, me pongo mal, y me resulta difícil dejar de pensar en eso
Me invento preocupaciones, sin sentido, sobre posibles enfermedades o gérmenes

Bibliografía

[1] Carbonell, J. G. "Derivatinal Analogy: A Theory of Reconstructive Problem Solving and Expertice Acquisition". Machine Learning. Vol 2. Cap 14. pp 371-392. 1986.

[2] De Raedt, L.; Beruynooghe, M. "Interactive concept-learning and constructive induction by analogy". Machine Learning. Vol 8. pp 107-150. 1992.

[3] Burstein, M. H. "Concept formation by incremental analogical reasoning and debugging". Machine Learning. Vol 2. Cap 13. pp 351-370. 1986.

[4] Greiner, R. "Learning by Understanding Analogies". pp 1-28. Computer Science Departament. University of Toronto.

[5] Simon, H. A. "Why Shoud Machines Learn". Machine Learning. Cap 2. pp 25-39. 1983.

[6] Dietterich, T.; Michalsky, R. " A Comparative Review of Selected Methods for Learning from Examples". Machine Learning. Cap 3. pp 41-83. 1983.

[7] Quinlan, J. R. "Induction of Desicion Tree". Machine Learning. Vol 1. pp 81-106. 1986.

[8] Quinlan, J. R. "Learning Efficient Classification Procedures and their Applications to Chess and Games". Machine Learning. Cap 15. pp 463-485. 1983.

[9] Utgoff, P. E. "Shift of bias for inductive concept learning". Machine Learning. Vol 2. Cap 5. pp 107-148. 1986

[10] Michalski, R. S. "A Theory and Methodology of Inductive Learning". Artificial Intelligence. Vol 20. pp 111-116. 1963.

[11] Carbonell, J. G. "Learning by Analogy: Formulating and Generalizing Plans from Past Experience". Machine Learning. Cap 5. pp 135-162. 1983.

[12] Michalski, R. S. "Understanding the nature of learning: issues and research dir Machine Learning. Vol 2. Cap 1. pp 3-26. 1986.

[13] Winston, P. H. "Learning by augmenting rules and accumulating censors". Machine Learning. Vol 2. Cap 3. pp 45-61. 1986.

[14] Dietterich, T. G.; Michalski, R. S. "Learning to predict sequences". Machine Learning. Vol 2. Cap 4. pp 63-106. 1986.

[15] Lebowitz, M. "Concept learning in a rich input domain: generalization-based memory". Machine Learning. Vol 2. Cap 8. pp 193-214. 1986.

[16] Sammut, C; Benerji, R. B. "Learning concept by asking questions". Machine Learning. Vol 2. Cap 7. pp 167-191. 1986.

[17] Carbonell, J.; Michalsky, R; Mitchell, T; "An overview of machine learning". Machine Learning. Cap 1. pp 3-23. 1983.

[18] Lenat, D. "The role of heuristics in learning by discovery: three case studies". Machine Learning. Cap 9. pp 243-306. 1983.

[19] DeJong, G. "An approach to learning from observation". Machine Learning. Cap 19. Vol 2. pp 571-590. 1986.

[20] Becker J. "AQ-PROLOG: A Prolog Implementation of an Attribute-Based Inductive Learning System". Reports of the Intelligent Systems Group. Cap 2. pp 1-5. 1985.

[21] Michalsky, R; Stepp, R. "Learning from observation: conceptual clustering". Machine Learning. Cap 11. pp 331-363. 1983.

[22] Langley, P; Zitkow, J; Simon, H; Bradshaw, G. "The search for regularity: four aspects of scientific discovery". Machine Learning. Cap 16. Vol 2. pp 425-469. 1986.

[23] Haas, N; Hendrix, G. "Learning by being told: acquiring knowledge for information management". Machine Learning. cap. 13. pp 405-427. 1983.

[24] Mostow, D. "Machine transformation of advice into a heuristic search proedure". Machine Learning. Cap 12. pp 367-403. 1983.

[26] Kolodner, J. "What is case-based reasoning". Case-based reasoning. Cap. 1. pag. 3-31. 1993.

[27] Kolodner, J. "Case studies of several case-based reasoners". Case-based resoning. Cap. 2. pag. 33-61. 1993.

[28] American Psychiatric Association. "DSM-III". 1983.