UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS

TESIS DOCTORAL

DESARROLLO DE UNA ARQUITECTURA CBR PARA LA RESOLUCIÓN DE TAREAS BASADAS EN EL PENSAMIENTO ANALÓGICO INTRADOMINIO

ABRAHAM RODRÍGUEZ RODRÍGUEZ

Las Palmas de Gran Canaria, 1996

54/1995-96

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
UNIDAD DE TERCER CICLO Y POSTGRADO

Reunido el día de la fecha, el Tribunal nombrado por el Excmo. Sr. Rector Magfco. de esta Universidad, el/a aspirante expuso esta TESIS DOCTORAL.

Terminada la lectura y contestadas por el/a Doctorando/a las objeciones formuladas por los señores miembros del Tribunal, éste calificó dicho trabajo con la nota de APTO CUM LAUDE
Las Palmas de Gran Canaria a 26 de Junio de 1996.

El/a Presidente/a: Dr. D. Roberto Moreno Díaz,

El/a Secretario/a: Dr. D. Roberto Moreno Díaz,

El/a Vocal: Dr. D. Enric Plaza Cervera,

El/a Vocal: Dr. D. José Luis Freire Nistal,

El/a Vocal: Dr. D. Roque Marín Morales,

El Doctorando: Dr. D. Abraham Rodríguez Rodríguez,

Universidad de Las Palmas de Gran Canaria

Doctorado en Informática

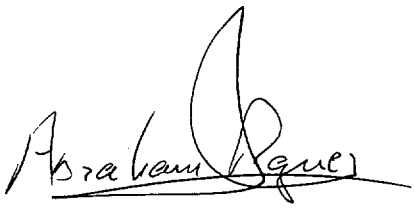
Departamento de Informática y Sistemas
Programa de Percepción Artificial y Aplicaciones

DESARROLLO DE UNA ARQUITECTURA CBR PARA LA RESOLUCIÓN DE TAREAS BASADA EN EL PENSAMIENTO ANALÓGICO INTRADOMINIO

Tesis Doctoral presentada por D.
ABRAHAM RODRÍGUEZ RODRÍGUEZ

Dirigida por el Dr. D.
FERNANDO MARTÍN RUBIO

El Director



El Doctorando

Las Palmas de Gran Canaria a 25 de Abril de 1996

AGRADECIMIENTOS

En primer lugar, quiero hacer constar mi agradecimiento al Doctor D. Fernando Martín Rubio, director de esta Tesis, por su extraordinario interés y apoyo para la realización de este trabajo.

También, al grupo de investigación de Ingeniería del Software y del Conocimiento por el apoyo que me han dado todos sus miembros, y especialmente a José Tomás Palma Méndez por la colaboración proporcionada durante la realización de algunas de las pruebas.

Asímismo, quisiera mostrar mi reconocimiento a D. Luis Álvarez León, quien resolvió mis dudas matemáticas con claridad y sencillez.

Quiero agradecer a los componentes del Departamento de Informática y Sistemas de la Universidad de Murcia la cordialidad y hospitalidad con la que me acogieron durante mi estancia en esa tierra, posibilitando que mi trabajo fuera lo más efectivo posible.

A un montón de amigos y compañeros que me han demostrado su interés durante todo este tiempo.

Y por último, y no por ello menos importante, vaya mi agradecimiento a Francisca C. Quintana, sin cuya confianza y subjetividad dudo que este trabajo viera la luz.

A mis Padres

Introducción

El Razonamiento Basado en Casos (CBR) utiliza experiencias pasadas para entender y resolver nuevos problemas [KOL92]. Cuando se presenta una nueva situación, el sistema 'recuerda' situaciones anteriores similares y adapta sus soluciones para que se verifiquen las nuevas restricciones. Esta es la forma usual de inferencia en muchos casos de la vida diaria; solemos ir a restaurantes en los que hemos estado previamente porque nos gustó el ambiente, la comida, el servicio, etc. Un especialista en diseño de cualquier tipo de componentes suele reutilizar diseños parciales que se han demostrado válidos en el pasado. Estas situaciones se producen con muchísima frecuencia aunque nosotros no seamos conscientes de ellas. Autores como Schank [SCH82; SCH86] han intentado modelizar los mecanismos que utilizamos cuando traemos experiencias pasadas a la conciencia, y dichos trabajos son los que han servido como base a los sistemas CBR.

Los sistemas CBR surgen como una alternativa a los sistemas tradicionales basados en reglas (RBS). Al igual que sucedió durante la década de los setenta y parte de los ochenta con los RBS estamos en una fase de euforia con los sistemas CBR. Este entusiasmo disminuirá o continuará según sea el éxito que tengan las aplicaciones comerciales que se están desarrollando en este momento.

Como cualquier alternativa, los CBR presentan una serie de ventajas respecto a los tradicionales RBS [KOL87; VAR93]: primero, rememorar (o recordar) errores previos avisa al sistema de resolución de problemas de qué errores se cometieron, evitando que se vuelvan a cometer los mismos errores. Segundo, estos sistemas no necesitan tener un conocimiento profundo del dominio sobre el que desarrollan la aplicación, aunque de tenerlo mejorarían el rendimiento. Tercero, las soluciones no tienen que desarrollarse desde cero ya que se parte de situaciones similares que comparten algunas decisiones que ya han demostrado su validez, y que pueden ser reutilizadas con ciertas garantías de éxito. Esto permite que se reduzca el espacio de búsqueda a la vez que se simplifica el proceso de satisfacción de restricciones. Cuarto, si se pudieran derivar esquemas abstractos de datos a partir de los casos conocidos, el conocimiento generalizado podría mejorarse. Decisiones que necesitaban muchos pasos de razonamiento pueden ahora hacerse a partir de la aplicación de esquemas generalizados, los cuales actúan como justificaciones de las decisiones que toma el sistema. Quinto, la fase de adquisición del conocimiento para las aplicaciones CBR es un paso mucho menos costoso, ahorrando gran cantidad del esfuerzo que se venía dedicando en el desarrollo de los sistemas tradicionales. Estas son las ventajas que usualmente se achacan a los sistemas CBR sin una

mayor discusión. Pensamos que algunas de ellas deben ser matizadas y valoradas en su justa medida, tarea que realizaremos a lo largo de la presente memoria.

Con el siguiente ejemplo queremos poner de manifiesto el ciclo de consulta de un sistema CBR, a la vez que ilustrar la potencialidad de este paradigma.

Supongamos que un cliente (X) se acerca a una oficina bancaria para solicitar un préstamo. Tras cumplimentar la solicitud correspondiente tendrá lugar una entrevista con el empleado del banco que es el que decidirá sobre la concesión o no del crédito requerido. En este caso se trata de un individuo de 27 años que solicita un préstamo para la compra de un coche con un plazo de amortización de 18 meses. Posee un trabajo temporal que en principio acaba antes de que se pueda pagar el crédito.

El primer paso que realiza el empleado es el de comprobar la solvencia del cliente, y confirmar que realmente podría pagar el crédito. Utiliza una fórmula tal que los ingresos menos los gastos fijos que tiene el cliente, menos la cuota del préstamo sea mayor que un cierto margen que suele rondar el 20% de los ingresos fijos. El empleado del banco desconfía de los contratos temporales, por lo que en principio es reacio a la concesión del crédito. Sin embargo, es consciente de que a una persona relativamente joven difícilmente se le puede exigir un trabajo permanente. También recuerda una situación similar en la que un cliente (Y) presentaba unas características laborales similares. En aquella ocasión se le concedió el crédito porque Y demostró un cierto interés por los asuntos relacionados con la empresa en la que trabaja (solvencia de la empresa, posición frente a la competencia, etc...). Además, se modificó el plazo de amortización de la deuda de forma que pudiera pagar el préstamo antes de que su contrato finalice. El empleado sondea la opinión de X sobre su empresa para comprobar si realmente se interesa por su trabajo (y por su supuesto interés en cumplir su contrato). Tras verificar el interés de X, modifica el plazo de amortización de 18 a 12 meses, y vuelve a comprobar el balance del cliente con la nueva cuota resultante. En esta ocasión el resultado del balance es menor del 20% exigido aunque mayor que cero. Ante esta situación lo usual es solicitar un aval del cliente. Finalmente se concede el crédito al cliente con un plazo de amortización de 12 meses, más la presentación de un avalista.

En este escenario hipotético se están utilizando ejemplos y contraejemplos para decidir sobre la concesión o no de un préstamo a un banco. Se está utilizando Razonamiento Basado en Casos. En CBR el individuo recuerda situaciones previas, similares a la actual para sugerir una forma de resolver el nuevo problema, de adaptar una solución que no se ajusta completamente (reducir el plazo de amortización), de prevenir posibles problemas (impago por falta de solvencia), y de buscar alternativas a los problemas detectados (solicitar un avalista).

Como una primera aproximación, los pasos por los que debería pasar un sistema que pretendiese automatizar el comportamiento anterior se muestra en la figura 1.

Con el primer paso se definen las características de la nueva situación que se pretende resolver. Utilizando estas características como índices se recupera de la memoria de casos aquel episodio que mejor represente la situación actual. Este caso recuperado será finalmente

adaptado para que su solución pueda verificar las restricciones impuestas por el problema actual.

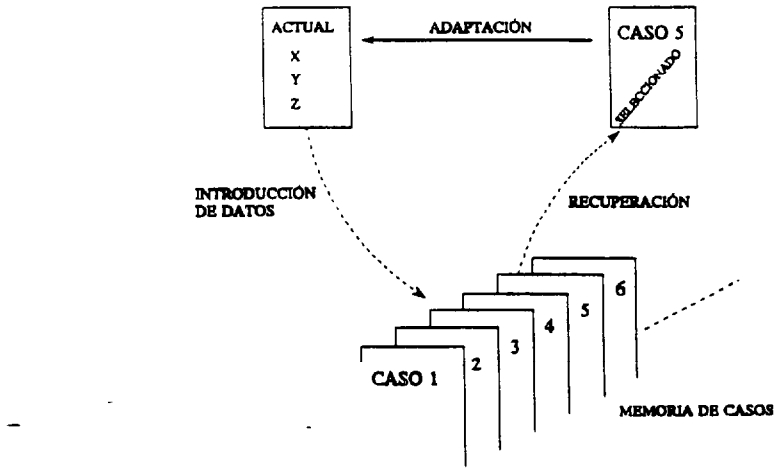


Figura 1: Ciclo de consulta básico

El esquema anterior es claramente insuficiente, sobre todo si queremos resolver problemas con una complejidad mediana como la del ejemplo. Para que el sistema sea más potente, es necesario proporcionarle una serie de estructuras y mecanismos que le permitan realizar inferencias y construir soluciones complejas, como la que refleja la figura 2.

Cuando se realiza una nueva consulta, las características del problema vienen definidas por una serie de datos de bajo nivel (datos concretos como ingresos mensuales, crédito solicitado, etc...). Sobre estos datos de entrada, el sistema realiza una serie de elaboraciones para derivar nuevos datos que podrán ser útiles durante la resolución (el balance lo puede calcular el sistema como una simple resta: ingresos - gastos mensuales, o codificando una jerarquía de conceptos, una elaboración consistiría en la abstracción de alguna característica: un duplex es un tipo de vivienda) [LEA92]. Los casos que existen en memoria no tienen porqué ser casos reales. Pueden haberse desarrollado en razonamientos anteriores aunque nunca hayan ocurrido. Pueden ser composiciones de casos, estereotipos, o casos específicos totalmente simulados.

La recuperación es uno de los mecanismos más complejos de un sistema CBR. Aunque algunos autores recuperan un sólo caso, normalmente se recuperan varios candidatos [SIM89; SIM91]. Estos posteriormente entrarán en un proceso de validación hasta que sólo quede un vencedor.

Una vez seleccionado un caso ¿qué se hace con él? Kolodner apunta algunas posibilidades en [KOL88]:

1. *Transferir la solución que alcanzó la meta en el caso anterior.*
2. *Transferir la solución que alcanzó la meta y modificarla basándose en las diferencias entre el caso actual y el anterior.*
3. *Transferir el método de inferencia con el que se alcanzó la meta en el caso anterior.*

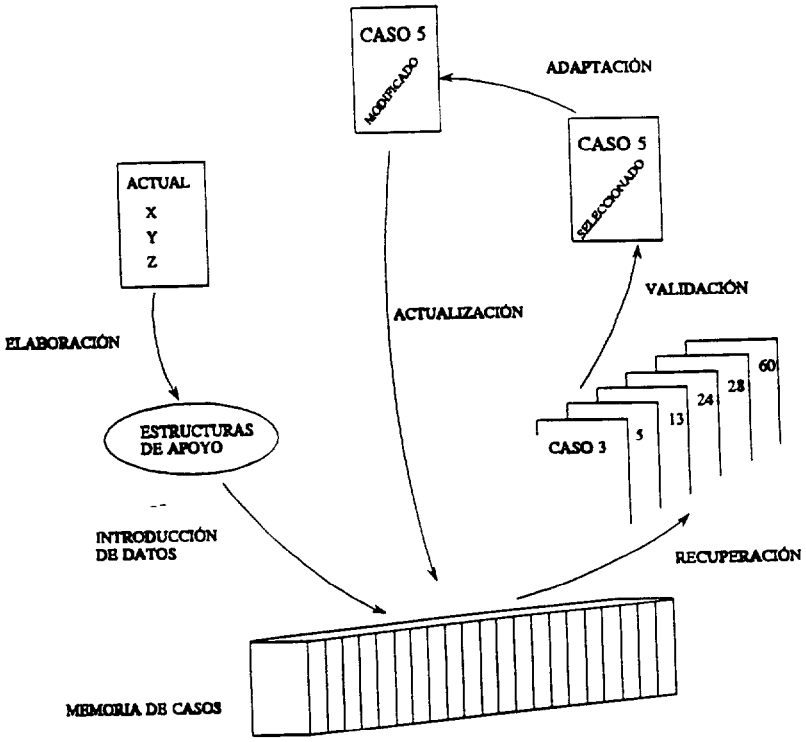


Figura 2: ciclo de consulta extendido

Idealmente, el sistema debe complementarse con un módulo de comprobación y otro de evaluación del resultado de la adaptación, realimentando el proceso en el caso de que encuentre alguna anomalía en la solución. Este mecanismo es el que permite anticiparse y evitar futuros errores similares al actual.

Finalmente, se procede a la actualización de la base de casos. No consiste en incluir cada situación resuelta en la base de casos, ya que se produciría un crecimiento incontrolado de la misma, degradándose de esta manera el rendimiento del sistema. Este problema puede evitarse, al menos parcialmente, si el sistema es capaz de organizar eficientemente sus estructuras de datos y enlaces, detectando y eliminando de la base aquellos casos que se vuelvan obsoletos, y formulando generalizaciones que agrupen a un conjunto de situaciones más o menos similares.

Según estemos aplicando el esquema anterior (u otro similar) a un problema en su conjunto, o sobre determinadas secuencias de la inferencia utilizada para resolverlo, estaremos hablando de Analogía Transformacional o Analogía Derivacional [CAR86]. Hay que resaltar la ventaja de esta última en los sistemas de planificación, donde es posible aprovechar segmentos (determinadas partes del plan antiguo) sobre nuevas situaciones. En esta situación se producirán varios cambios de contexto a causa de que el centro de atención cambie de un caso a otro, por lo que es necesario complementar el sistema con técnicas de razonamiento no-monótono o con algoritmos de Mantenimiento de la Realidad (ej.: TMS de Doyle [SHA89]). De esta manera se garantiza la consistencia entre las hipótesis que se vayan generando.

Aunque suele existir un cierto recelo hacia la utilización del término *Analogía*, los sistemas CBR resuelven realmente problemas analógicos por cuanto utilizan conocimiento episódico para la resolución de problemas. Ahora bien, los CBR han demostrado realmente su eficacia a la hora de atacar problemas analógicos siempre dentro de un mismo dominio, y no tanto en problemas en los que se ven implicados múltiples dominios. En el primer caso estaremos hablando de analogía intradominio, y en el segundo de interdominio. Desde este punto de vista, los CBR constituyen tan sólo una de todas las posibles maneras de llevar a cabo tareas analógicas intradominio, y, en nuestra opinión, la que más se acerca a la forma que utilizamos los humanos cuando resolvemos muchas de las tareas complejas.

El Razonamiento Basado en Casos deriva de una visión del entendimiento como un proceso de justificación [SCH82]. Utilizamos nuestro conocimiento acerca de lo que está escrito o se dice cuando leemos un texto o atendemos en una discusión, para poder unir las piezas de lo que leemos. Nuestro conocimiento nos ayuda a predecir lo que viene a continuación, a eliminar la ambigüedad en las palabras, y a conectar la variedad de cosas que se dicen. Schank propuso que nuestro conocimiento acerca de situaciones se almacena en *scripts*, y éstas nos permiten establecer expectativas de lo que vamos a escuchar, lo que de hecho, nos permite inferir las relaciones entre las cosas que oímos. Pero los scripts sólo proporcionan un tipo de conocimiento, y nosotros también usamos conocimiento acerca de metas, planes, relaciones interpersonales, caracteres, etc.. El *entendimiento* se concibe como un proceso de aplicación del conocimiento en el que se construyen justificaciones que enlazan las piezas individuales de un discurso.

Esta visión del entendimiento afirma que cuando están disponibles las estructuras de conocimiento adecuadas, el entendimiento se convierte en un proceso top-down relativamente simple. Las estructuras de conocimiento nos permiten generar expectativas, permitiéndonos reconocer como encaja algo nuevo en el resto de la historia. La principal aportación de Schank radica en que especifica muchos tipos de estructuras de información útiles y propone los mecanismos de inferencia para poderlas utilizar (p.e. las metas y los planes se gestionan por un mecanismo que es capaz de predecir qué planes son aplicables cuando una persona quiere alcanzar una determinada meta).

La primera vez que oímos algo nuevo, utilizamos una gran cantidad de recursos cognitivos para unir y relacionar las piezas del discurso. Sin embargo, la segunda vez rememoraremos aquella primera, que nos proporciona las expectativas (scripts) que nos permitirán interpretar la historia de forma top-down, reconociendo algo que pasa como algo que esperamos, en lugar de tener que atar las piezas desde cero. El proceso de

rememoración utiliza tanto estructuras generales de conocimiento como instancias específicas (casos). Por lo tanto, deben existir mecanismos que nos permitan recuperar y usar ambos tipos de conocimiento.

Una cuestión adicional es: ¿Cómo se encuentra y se recupera el conocimiento adecuado?. La clave radica en la *indexación*, identificada por Kolodner [KOL93a]. Las estructuras de memoria están etiquetadas de acuerdo con su tipo y sus distinciones con otras estructuras similares. Si tanto las estructuras generales y los casos están etiquetados o indexados según lo que las diferencia de otras estructuras similares, entonces ambas serán igualmente accesibles por los mismos mecanismos de recuperación.

La teoría *Dinamic Memory* [SCH82] considera que la rememorización, la comprensión, la experiencias vividas y el aprendizaje no pueden separarse unas de otras. Nuestra memoria, que es dinámica, cambia como resultado de nuestras experiencias debidas a las nuevas cosas que nos encontramos, las cuestiones que surgen en nuestra mente como resultado de esas experiencias, y la forma en que respondemos dichas cuestiones. Las experiencias proporcionan las expectativas que permiten al proceso de comprensión ser simple y top-down. Las experiencias individuales y las descripciones generales de las experiencias normativas se encuentran a medida que las entendemos; ambas proporcionan los mismos tipos de información para los procesos de comprensión. El efecto colateral de este proceso es que la memoria dinámica nunca se comporta dos veces exactamente de la misma manera porque cambia como resultado de cada una de sus experiencias. El proceso de razonamiento nunca encuentra exactamente el mismo conocimiento en dos barridos de la memoria.

A pesar de lo dicho anteriormente, en el sentido de que el entendimiento es un proceso que combina la utilización de los casos y el conocimiento general, prefiriendo aquel que proporcione las expectativas más específicas, esto no se ha visto posteriormente reflejado en el desarrollo del área de CBR. La mayoría de las investigaciones favorecen los casos en perjuicio del conocimiento general. ¿por qué?. Kolodner [KOL93b], aporta un motivo pragmático. En la resolución de problemas, al igual que durante los procesos de entendimiento, las predicciones más útiles son usualmente las más específicas. Queremos que nuestros procesos de resolución hagan uso del conocimiento más específico disponible. A menudo, esto lo proporcionan los casos.

Por lo tanto, la representación de la información es, y ha sido, uno de los primeros puntos de divergencia en la investigación en este campo. Frente a la mayoría de sistemas en los que se potencia la estructuración de la información en casos de bajo nivel, nos encontramos otros en los que coexisten episodios generalizados (MEDIATOR [KOL89], JULIA [HIN91], CELIA [RED90]). Decidirse por un modelo de representación del conocimiento debe realizarse en coordinación con los otros aspectos que hay que considerar en el diseño de un sistema de estas características, que son, entre otros: a) los mecanismos de coordinación entre las estructuras representacionales implicadas; b) la indexación de los casos de memoria, o qué índices o características se utilizan para clasificar los casos y cómo se utilizan para facilitar la recuperación de los mismos; c) las técnicas que se utilizan para medir la similaridad a nivel global entre dos situaciones y, a nivel local, entre dos características; d) el método utilizado para adaptar una solución previa a las circunstancias actuales; e) o la asimilación de nuevas experiencias y la consiguiente reorganización de la base de conocimiento. Watson y Marir [WAT94], y Aamodt y Plaza [AAM94] realizan un muy buen recorrido introductorio por cada uno de estos aspectos.

Por otra parte, la especificación de una arquitectura de CBR u otra depende en gran parte de la tarea que el sistema tenga que resolver. Los siguientes párrafos pretenden comentar algunas de las tareas en las que el uso de esta 'tecnología' ha tenido un éxito apreciable. Una revisión más extensa puede encontrarse en el artículo de Kolodner [KOL92], en la publicación [ALTH95], o en el trabajo [ROD94].

En los problemas de *diseño*, donde estos se definen en función de un conjunto de restricciones, el sistema debe suministrar un medio para resolver dichos problemas. Una restricción se plantea normalmente como un condicionante sobre los valores que pueden tomar al menos una pareja de atributos del sistema. Usualmente, las restricciones no definen completamente el problema (permiten la existencia de demasiadas soluciones), o imponen demasiadas condiciones (no existe ninguna solución). En este último caso, la resolución del problema requiere la especificación del problema de forma que se verifiquen las restricciones más importantes, y siendo las otras negociables.

En CBR, los casos en memoria proporcionan soluciones globales a problemas globales. Los problemas no se descomponen y se resuelven independientemente, sino que se adopta la solución global del caso en memoria. Aquellas partes que no se ajusten perfectamente a la nueva situación tendrán que ser modificadas.

Resolver un problema adaptando una solución pasada evita tener que controlar simultáneamente todas las restricciones, y tener que descomponer el problema en subproblemas para posteriormente recomponerlos. La verificación de un conjunto *cerrado* de restricciones en el caso en memoria libera al sistema de tener que volver a comprobar dichas restricciones cuando la situación actual posee el mismo conjunto de restricciones o un subconjunto del mismo. Es decir, cuando adoptamos un caso de memoria también adoptamos las decisiones que se tomaron cuando se resolvió dicho caso, y en la situación actual ya no es necesario tener que rehacerlas. Lo que sí es necesario realizar es comprobar la consistencia de dichas decisiones con los datos actuales. Para ello se utiliza normalmente un sistema de mantenimiento del razonamiento.

Normalmente será necesario más de un caso para resolver un problema de diseño. En general, algunos casos proporcionan una estructura (esqueleto) para la solución y se utilizan otros casos para completar los detalles.

Algunos sistemas de diseño desarrollados utilizando CBR son JULIA [HIN89; KOL87] que compone menús gastronómicos, o CLAVIER [BARL88] que está siendo utilizado por la Lookheed para distribuir las piezas de aleaciones de materiales dentro de un horno.

La planificación consiste en la especificación de una secuencia de pasos con el objeto de alcanzar algún estado del mundo determinado a priori. Existen múltiples problemas asociados con la planificación. Los buenos planes son secuenciales siempre que es posible, de forma que los últimos pasos del plan no deshacen los resultados de los primeros y de forma que las precondiciones de los últimos pasos del plan no son violadas por los resultados de los pasos anteriores. Otro problema son las precondiciones. Un planificador debe estar seguro de que se verifican las precondiciones de cualquier paso del plan antes de especificar dicho paso. Por lo tanto, la planificación implica la especificación de los pasos que permiten verificar las precondiciones, además de la especificación de los pasos propiamente dichos. Un (sub)plan se podría considerar como un procedimiento, y las precondiciones como los parámetros de entrada al mismo. Así, no podríamos llamar al procedimiento hasta que no dispusiéramos de los valores necesarios.

El razonamiento basado en casos afronta dichos problemas proporcionando planes que se han usado en el pasado y han demostrado su eficacia. El planificador sólo tiene que realizar cambios menores en el plan en lugar de tener que construirlo desde cero. Los cambios suelen consistir en recombinaciones de los pasos del propio plan,

sustituciones por pasos similares que verifican las mismas restricciones y objetivos, o eliminaciones de pasos que no tienen sentido en la nueva situación.

El ejemplo más representativo es el sistema CHEF [HAM89], el cual diseña y planifica la preparación de nuevas recetas de cocina.

Los sistemas de *diagnosis* proporcionan una explicación a un conjunto de síntomas dado. Poseer una base de casos resueltos simplifica la tarea de tener que reconstruir las cadenas causales en cada consulta a la base, adaptando si fuera necesario la diagnosis antigua a la nueva situación.

Por ejemplo, CASEY [KOT88] es capaz de diagnosticar problemas cardiacos mediante la adaptación de problemas similares de otros pacientes.

Los sistemas que intentan *justificar* el comportamiento de determinados sucesos utilizan episodios similares de los que 'toman prestadas' las justificaciones o explicaciones que se aplicaron en dichas circunstancias.

La necesidad de una justificación surge cuando la situación que se está observando no se ajusta al comportamiento usual en el dominio, es decir, se ha detectado una anomalía. Una justificación es una cadena causal que demuestra porqué se ha producido el comportamiento anómalo mediante la especificación de un conjunto de premisas que causalmente conducen a dicha proposición. [RAM93]

Un sistema que trabaje con explicaciones debe ser capaz de detectar anomalías, y de resolverlas mediante la construcción de justificaciones causales para los eventos en la historia, de forma que pueda entender porqué los actores actuaron como lo hicieron.

Por ejemplo, el sistema AQUA [RAM93] construye explicaciones acerca de comportamientos suicidas de terroristas.

Por último, queremos centrar los objetivos de nuestro trabajo de investigación, al mismo tiempo que apuntar cuales van a ser los contenidos de los capítulos siguientes.

No hay mas que observar la gran cantidad de congresos internacionales que se organizan anualmente relacionados específicamente a los CBR, o comprobar la gran cantidad de artículos de CBR que aparecen en las revistas especializadas en Inteligencia Artificial para darse cuenta de la importancia y del empuje que está ganando esta tecnología. Nosotros compartimos las buenas expectativas de futuro que

diversos autores le dan a este campo [WAT94; KOL93b] y queremos incorporarnos en la medida de lo posible a esta corriente. Por otro lado, encontramos un vacío importante en la investigación relacionada con este campo en España. Únicamente hemos localizado a un grupo consolidado en el CSIC dirigido por Enric Plaza, y creemos que, a pesar de la excelente calidad de sus trabajos, es una aportación insuficiente para un potencial investigador como el de nuestro país. Por estas razones, y por el atractivo que tiene para nosotros esta tecnología, dirigimos nuestro esfuerzo hacia el desarrollo e implantación de una línea de investigación en CBR, a pesar del retraso significativo que llevamos respecto a otros grupos europeos y americanos.

El desarrollo de una nueva línea de trabajo debe realizarse de forma organizada y escalonada. Las tareas a realizar en un momento determinado deben sustentarse en el trabajo previo realizado, y una vez desarrolladas deben contrastarse para que otras tareas puedan apoyarse, a su vez, sin dudas acerca de su funcionalidad. Para conseguir nuestros objetivos, proponemos un plan compuesto de dos partes: en la primera, se establecen los fundamentos de la línea de trabajo, y en la segunda, se consolidan los resultados permitiendo una expansión de la investigación en múltiples direcciones.

Establecer los fundamentos de la línea de trabajo persigue fijar las bases sobre las que descansará todo el trabajo futuro, por lo que es importante el análisis y la justificación previa de las decisiones importantes. En principio, el punto natural de partida es el diseño de una nueva arquitectura de sistema CBR, que recoja las principales características de los mejores sistemas actuales y que, simultáneamente, elimine sus principales defectos. Hemos descartado la adopción de un sistema ya desarrollado por múltiples motivos, aunque principalmente porque sólo el desarrollador conoce las verdaderas razones de las decisiones de diseño del sistema, y, por lo tanto, la funcionalidad real del mismo. El análisis previo lo desarrollamos en el capítulo 1, y de este obtendremos las principales características que deberá tener nuestro sistema.

Una vez diseñado, es necesario desarrollar su implantación progresiva desde una doble perspectiva funcional. Por un lado, se limita conscientemente la complejidad de los problemas que será capaz de afrontar, con miras a una simplicidad mayor de las estructuras de conocimiento implicadas. Y por otro, se limitan las capacidades de procesamiento y de representación para poder ir comprobando su funcionamiento incrementalmente. Así, en una primera fase, el sistema poseería una capacidad funcional limitada a resolver problemas de diagnosis relativamente sencillos, utilizando unas características de razonamiento analógico acordes con el problema que resuelven. En fases sucesivas se aumentará la complejidad de los problemas a los que se enfrente el Sistema, a la vez que se le dota de los mecanismos necesarios para poderlos

resolver. Para esta segunda fase, el sistema debe ser capaz de realizar tareas de diagnosis compleja, utilizando los principales recursos de los que dispondrá el sistema definitivo. Además, consideramos que este debe ser nuestro objetivo para el presente trabajo.

Antes de que la línea pueda ser totalmente consolidada deben comprobarse y contrastarse los resultados obtenidos hasta este momento, mediante pruebas sistemáticas utilizando diversos dominios de trabajo. Sólo entonces se podrá expandir el Sistema para que refleje un comportamiento dinámico, con capacidades plenas de adaptación y de aprendizaje. Estos dos tópicos son los que actualmente están siendo desarrollados por los grupos de investigación que llevan más tiempo trabajando en este área. Esto es así porque las técnicas de adaptación y aprendizaje sólo pueden aplicarse a arquitecturas suficientemente contrastadas, por lo que no puede ser un objetivo prioritario en la primera fase de implantación de la línea de investigación.

El capítulo I desarrolla un análisis profundo de los sistemas desarrollados hasta la fecha, del cual derivaremos las principales carencias presentes en el área, y que están recogidas en la primera sección del siguiente Capítulo.

Desarrollamos la propuesta de una nueva arquitectura para un sistema CBR en el Capítulo II, aunque centrándonos exclusivamente en su especificación formal. La definición del Sistema se realizará de manera incremental, empezando por los componentes primitivos, y construyendo sobre estos las definiciones más complejas.

En el Capítulo III discutimos las principales decisiones de diseño que se tomaron durante la realización del trabajo. Hemos agrupado las decisiones relacionadas con las estructuras de representación e inferencia, las relacionadas con la función de similaridad, con diversos aspectos del funcionamiento del Sistema, y con el ajuste de la función de similaridad. Finalizamos este Capítulo comentando el comportamiento del sistema en distintas situaciones y como éstas afectan al rendimiento del mismo.

El último capítulo recoge las principales conclusiones a las que hemos llegado tras la realización de este trabajo, comentando las características más sobresalientes, limitaciones, posibilidades de expansión, así como las conclusiones propiamente dichas.

Capítulo I

Análisis de los Sistemas de Razonamiento Basado en Casos

En el Capítulo anterior introdujeron los principales aspectos relacionados con los Sistemas de Razonamiento Basado en Casos. Este capítulo analizará en profundidad los principales trabajos desarrollados hasta la fecha tomando dichos aspectos como referencia organizativa. El objetivo es el de conseguir una visión lo más completa posible de las distintas técnicas que se han venido utilizando en cada uno de los temas apuntados. Este estudio será el que nos permitirá derivar cuales son las áreas con problemas no resueltos y que pueden ser objeto de investigación en capítulos posteriores.

Contenidos

1. Representación de casos: estructura de los sucesos
2. Algoritmos de indexación: enlaces y recuperación
3. Vocabulario de indexación: qué características considerar
4. Métrica de similaridad: cuando son dos casos similares
5. Adaptación de casos: qué y cómo adaptar a la nueva situación

1 Representación y Organización de la Memoria

Ya se comentó anteriormente que los sistemas CBR desarrollados hasta la fecha estaban muy influenciados por los distintos requisitos funcionales que imponían las diferentes áreas de aplicación. Cuando se estudian las estructuras representacionales que utilizan estos sistemas, este hecho queda patente; se pueden encontrar desde casos representados simplemente por listas de características hasta complejas descripciones causales del proceso de razonamiento justificadas detalladamente.

El trabajo más simple de los revisados es el de la herramienta de desarrollo creada por Inference Co, CBR-Express [CBR91], la cual utiliza un conjunto limitado de características organizadas en una base de datos relacional. Esta estructura sólo es util para tareas de diagnosis. Además, sus algoritmos presentan una serie de deficiencias como el prerrequisito de que las características implicadas deben poseer una estructura uniforme (no presenta ningún tipo de elaboración o abstracción de la información de entrada).

La aproximación más intuitiva la constituyen las aplicaciones que intentan apoyarse en la naturaleza causal que tienen la mayoría de los dominios de investigación. Para ello se construyen casos a partir simplemente de listas de características. Dos ejemplos son CASEY [KOT88] y BATTLE PLANNER [GOO89]. De los dos sistemas, el más claramente causal es CASEY. Este sistema fue diseñado para detectar anomalías cardiacas a partir de casos codificados como una lista de 40 descriptores (edad, respiración, temperatura, ...). El sistema también estructura las relaciones (causales) entre entidades mediante probabilidades que indican el grado con el que un determinado estado puede deducirse de otro. El sistema contempla más de 140 estados y 100 relaciones. En la misma línea que CASEY, Battle Planner estructura su memoria con listas de características (en este caso 57). Cada lista registra el conocimiento particular de alguna de las 500 batallas que contiene el sistema mediante registros que indican el número de atacantes, de defensores, número de artillería, tipo de terreno, etc... Su objetivo es el de valorar la situación que se presenta en una nueva batalla terrestre. La diferencia con CASEY radica en la forma de codificar el conocimiento específico del dominio. En esta ocasión existen conceptos que podrán descomponerse en distintos niveles según sea el nivel de detalle con el que se precise razonar. Para ello el Battle Planner implementa una jerarquía simbólica en la que es posible descomponer, por ejemplo, los distintos tipos de terreno sobre los que se desenvuelve una batalla.

GREBE [BRAN91] y CABARET [RIS91] incluyen también conocimiento causal en forma de reglas como apoyo al sistema CBR. Ambos sistemas surgen al notarse que el sistema

diseñado por Koton no era capaz de hacer frente a nuevas situaciones que incluyeran varios casos de memoria simultáneamente. CABARET contiene un módulo de control que se encarga de pasar la ejecución entre el sistema de reglas y la base de casos. Este módulo se apoya en unas 36 heurísticas para tomar decisiones. Este sistema solo es aplicable a dominios en los que solo se necesiten hacer preguntas para encontrar argumentos a favor o en contra de una hipótesis que se suministra al sistema, sin que se necesite ningún tipo de elaboración de la información ni adaptación de los datos almacenados en los casos de la base. GREBE utiliza una estrategia totalmente distinta: por un lado, utiliza los datos de entrada para abstraer los antecedentes de las reglas, y por otro, utiliza las reglas para descomponer las metas de los casos de forma que se puedan verificar más fácilmente. Estos dos sistemas pueden funcionar correctamente para sistemas de interpretación y diagnosis siempre y cuando sean aplicaciones que no requieran de la no-monotonicidad, es decir, que se pueda llegar a una contradicción en la información que manejan. En estos casos, estos sistemas dejan de ser eficaces, ya que no almacenan ni gestionan las dependencias lógicas entre los conceptos del sistema.

La gran ventaja de utilizar jerarquías es que facilita el trabajo de generalización necesario para recuperar el caso adecuado a la nueva situación.

El sistema desarrollado por Simoudis [SIM91] estructura la base de casos a partir de generalizaciones de los mismos. Internamente un caso se representa como una lista de características de bajo nivel (sin elaborar) junto con dos descriptores especiales que son la validación y la solución del caso. La validación consiste en un conjunto de parejas de pruebas (tests) junto con los resultados de los mismos.

Las estructuras jerárquicas de conceptos son también la base del trabajo de Thompson con LABYRINTH [THO89]. En esta ocasión será la propia jerarquía la que aporte la estructura a la memoria de casos. Cuando lo que se tienen son conceptos compuestos, ya no es suficiente representarlos mediante parejas de valor-atributo. Cada caso se define como un conjunto de componentes relacionados con enlaces 'Parte-de'. Cada componente puede ser a su vez otro objeto estructurado o un componente primitivo (con atributos como color). En general, los nodos terminales se corresponden con casos específicos que se han observado. Los nodos no terminales representan un concepto probabilístico, el cual contiene una descripción de los casos almacenados debajo en la jerarquía. Un concepto es un conjunto de atributos y sus valores correspondientes. Asociado con cada valor está la probabilidad condicional de que dicho valor aparezca en la clase.

Para Alterman [ALTE89], los conceptos se corresponden con los distintos eventos que pueden surgir dentro de un caso, construyendo una red de coherencia de conceptos de eventos (ECC) en la que se pueden identificar relaciones entre los mismos de tres tipos: taxonómicas,

temporales, y partonómicas. La red ECC estructura una secuencia de eventos en la que las relaciones causales entre los eventos de un caso se representan en términos de una subred de ECC.

PRIAR [KAM89] también genera una jerarquía para la representación de los casos. Este es un sistema de generación de planes a partir de otros que ya posee en memoria. Mas que reutilizar viejos planes en parte o totalmente (para lo que sólo tendrían que verificarse las condiciones de aplicabilidad), de lo que se trata es de potenciar la flexibilidad de modificación. Es decir, se intenta reutilizar el proceso de planificación en lugar del resultado del mismo. Para ello no basta con sólo el plan almacenado; hay que incluir también el proceso de generación del plan y las interdependencias entre las distintas partes del proceso. Son las distintas tareas que componen el caso las que tienen la estructura jerárquica en el sistema. La red jerárquica de tareas muestra cómo el planificador comienza con una especificación abstracta del plan y la refina en un plan ejecutable, a través de sucesivas descomposiciones de las tareas. En los nodos se registra información sobre la validación de los distintos tipos de condiciones de aplicabilidad y de dependencias entre las distintas tareas en forma de conocimiento causal específico o interno al plan al que pertenece el nodo. Cada nodo representa sucesivos pasos del plan desarrollado, conteniendo el conjunto de condiciones necesarias para la validación de la parte del plan que sigue. Esta arquitectura precisa del desarrollo paralelo de un modelo causal del dominio de forma que se puedan adquirir mediante simulaciones la información de dependencias y validación.

La abstracción de casos mediante una jerarquía es común en muchos sistemas como el publicado por El-Gamal [ELG93], PERSUADER [SYC93], o ADAPtER [POR95]. Este último estructura la base de casos mediante una estructura de generalizaciones de casos en las que los nodos hoja se asimilan con los casos concretos, ya que estos nodos son los que tienen la información más específica. La abstracción se realiza en base a generalizaciones de las características que permiten diferenciar los unos de los otros. El sistema codifica distintas familias de índices que son específicos de la aplicación concreta (resolución de conflictos laborales) y que difícilmente son aprobechables o aplicables sobre otros dominios. En ADAPtER junto con las generalizaciones de casos, se utiliza un modelo causal que contiene por un lado las relaciones causales y, por otro, las relaciones entre estados y sus manifestaciones observables. Esta organización le permite ordenar los casos de memoria en función de su facilidad de adaptación a la situación actual mediante una función heurística. Por contra, requiere que el IC identifique la información a proporcionar en el caso entre datos contextuales y manifestaciones. Por otro lado, la función de similaridad heurística sumistrada no siempre es capaz de proporcionar un mejor caso, por lo que se vale de una segunda función de similaridad tradicional.

En general, todos los trabajos que pretendan razonar sobre algún tipo de eventos o sucesos

deben apoyarse en un *espacio de conceptos* sobre el que se puedan establecer generalizaciones, y a partir de estas adaptarse a la nueva situación. Si bien estos conceptos pueden estar definidos implícitamente dentro de la propia estructura de los casos como en los sistemas anteriores, algunos autores optan por definir un espacio paralelo de conceptos que coopere con la base de casos. Esta es la aproximación que siguió Sycara [SYC91], Leake [LEA91], o es el caso del TWEAKER [KAS88], un sistema que codifica los casos basándose en las teorías conceptuales de Schank. Cada caso es un XP (Explanation Pattern) en el que se relaciona un conjunto de hipótesis con sus consecuencias causales. Al igual que el sistema de Kambhampati lo se intenta capturar en los XP es el razonamiento que se siguió en la situación original. Todos los hechos que conoce el sistema se relacionan en una red semántica. Cada nodo es un concepto, y estos se relacionan unos con otros mediante relaciones de Parte-Subparte y otras de origen semántico. Ram [RAM93] también se ha apoyado en los XP para desarrollar su sistema AQUA. El mismo reconoce que el dominio sobre el que se aplica el sistema debe ser modelizable mediante patrones de causalidad representados mediante una estructura de grafo; es decir, deben identificarse todas las relaciones causales que existen en el dominio. Esto es así porque estos sistemas se apoyan (excesivamente) en un ajuste causal entre la situación nueva y las de memoria.

Los sistemas anteriores tienen una característica en común: de una forma u otra, consideran un caso como una estructura monolítica. Los más complejos permiten una descomposición interna, pero sólo para comprobar determinadas condiciones de aplicabilidad o permitir una mejor estructuración de las tareas del caso. Carbonell [CAR86] primero, y Kolodner con su proyecto JULIA posteriormente [KOL87], intentan sacar fruto de aquellas *partes* de los casos que realmente aportan algo a la nueva situación. Ambos autores se valen de trazas derivacionales para descomponer los casos y enlazar distintas partes de unos con otros. Para ello deben codificar todas y cada una de las decisiones que condujeron a la solución. Es decir, las alternativas consideradas, reglas de decisión, decisiones erróneas y sus justificaciones, secuencia de pasos, etc... Carbonell considera que codificando de esta manera el conocimiento episódico se evita la típica búsqueda sobre todo en las etapas iniciales. Además del conocimiento episódico, identifica otros tipos de conocimiento necesarios en este tipo de sistema:

Estático del dominio: se corresponde con la conceptualización del dominio y predicados. Los denomina hechos axiomáticos.
De trabajo del dominio: son las acciones del dominio (operadores, reglas de inferencia, heurísticas,..)
Problemas resolubles: es la especificación de los problemas que pueden resolverse con el conocimiento estático y de trabajo.

Sin embargo la propuesta de Carbonell posee un grave inconveniente. Cuando un caso que se está utilizando deja de ser válido, se busca otro de complemente al primero para reparar esa insuficiencia. Pero el nuevo caso recuperado deberá ser accedido desde el comienzo de la traza, y no desde el punto donde lo dejó el caso anterior. Realmente la analogía derivacional no recupera un caso con una submeta similar a la actual, sino que proporciona una forma alternativa de llegar hasta ella.

Kolodner va más allá, y además de la estructura derivacional de Carbonell incluye un sistema de mantenimiento de la verdad TMS. Este es útil en el caso en el que haya que retractar alguna de las decisiones hechas durante la consulta. El sistema TMS se vale de las cadenas de enlaces de justificación que construyen los casos, pero recorriéndolos en sentido contrario. En otro trabajo, Kolodner [KOL88] distribuye partes de los casos por jerarquías de abstracción que asocian con escenas del evento. Esta estructura permite construir generalizaciones sobre partes de los casos, gracias a los distintos niveles de abstracción en los que se reparten los sucesos de los casos. Esto permitiría, por ejemplo, obtener un conocimiento más específico con sólo descender por la jerarquía de abstracción. La propuesta de Kolodner se caracteriza porque no utiliza un caso 'guía' sino que va construyendo la nueva solución en base a segmentos de casos según los va necesitando. Este comportamiento exige la utilización de un sistema TMS, ya que debe verificar que cada segmento que se añade a la solución no rompe la consistencia del razonamiento. El (grave) inconveniente de este tipo de sistema radica en que es preciso identificar todas y cada una de las dependencias que existan entre los datos de forma que el sistema TMS las pueda mantener.

En contraste con esta última propuesta está también la de Redmon [RED90]. Este también descompone los casos en base al conjunto de hipótesis que lo conforman. Cada hipótesis es almacenada junto con el contexto que la define en unidades denominadas Snippets. Un caso se compone de un conjunto de Snippets. Esta estructura le permite reutilizar hipótesis parciales de casos alternativos cuando el caso que actualmente está proporcionando la línea de razonamiento es incapaz de verificar la hipótesis de la misma manera. Su sistema CELIA mejora sustancialmente las aproximaciones anteriores, sin embargo con el coste de mantener una compleja estructura de punteros y una redundancia (tal vez excesiva) de información en el sistema. Además, la representación del snippet no posibilita que pueda pertenecer a mas de un caso, ya que se mantiene dentro del snippet el contexto local que facilita la verificación de la hipótesis en cuestión. El sistema también viene condicionado por una declaración a priori del conjunto de hipótesis que componen cada caso, de forma que se puedan construir los snippets en base a los mismos. Tampoco queda muy clara la estrategia que debe seguir el sistema cuando debe decidir entre varios snippets de distintos casos que verifican la misma hipótesis, ya que en ningún momento se ordenan las alternativas.

Plaza [PLA95] también permite la reutilización de segmentos de casos para la

construcción de una nueva solución. Su propuesta consiste en la utilización de un lenguaje de representación centrado en el objeto denominado NOOS [LLU95], de forma que es posible representar objetos que se corresponden con casos, o con partes de casos. Las distintas partes de los casos se enlazan mediante la referencia de alguno de los atributos de un objeto del nombre de otro objeto. En definitiva, un caso viene definido por un grafo que se asemeja en cierto sentido a una estructura de Frames. Representa una forma novedosa tanto de representar el conocimiento como de valorar la similaridad (evitando la asignación de pesos).

Otra aproximación interesante es la de Blau [BLA91]. Este utiliza varias jerarquías para representar el conocimiento del Dominio además de una estructura de casos. Los conceptos se representan jerárquicamente mediante la relación 'es-un'. Las acciones poseen también su propia estructura jerárquica completada con una serie de enlaces de expectativas que codifican un conocimiento causal. Finalmente, las metas o planes se estructuran mediante su propio arbol AND/OR. La potencia de este esquema radica en su incompletitud. Es decir, no se precisa de un modelo completo del dominio para que el sistema sea operativo. La estructura interna de los casos está formada por un conjunto de parejas situación-solución. Las situaciones se componen de un estado inicial más las metas de alto nivel. La solución es una secuencia lineal de estados relacionados mediante inferencias causales con una red de eventos (relaciones acción-meta). La gran cantidad de tipos de conocimiento que deben codificarse con esta arquitectura (eventos, acciones, estados, interpretaciones, casos, planes, y conceptos) sitúan serias exigencias sobre las características del dominio que se pretende modelizar, a la vez que dificulta las tareas de adquisición y posterior formalización del conocimiento por parte del ingeniero del conocimiento.

Queda de manifiesto que es necesario codificar un conocimiento global a la base de casos. Algunos autores prefieren integrar este tipo de conocimiento (compuesto básicamente de conceptos) dentro de la base de casos ya sea generando cadenas causales entre las características de los casos, mediante fórmulas que los relacionen, o generando una estructura jerárquica *superpuesta* a la estructura de los casos. Otros sin embargo, utilizan un espacio paralelo en el que definen una red semántica que incluye a todos los conceptos que aparecen en el dominio. La propia estructura de los casos va desde una lista de características, pasando por estructuras complejas que si bien siguen siendo monolíticas, permiten registrar eventos mucho más complejos que los anteriores, hasta los sistemas que descomponen los ejemplares para posibilitar el aprovechamiento de aquella información que es realmente aplicable en la nueva situación.

2 Algoritmos de Indexación y Recuperación

Los algoritmos de indexación son los encargados de recuperar aquellos casos de memoria a partir de las características que mejor los representan. Decidir cuáles son estas características será el objetivo del próximo apartado (vocabulario de indexación). Durante las primeras épocas de los sistemas basados en casos se utilizaban simples redes de discriminación para organizar las estructuras de datos en memoria. Poco a poco, los autores fueron tomando conciencia de la importancia y de la complejidad del problema y se han empezado a desarrollar arquitecturas masivamente paralelas que minimizan los tiempos de acceso a la vez que son capaces de trabajar con estructuras de datos cada vez más complejas. Desde el comienzo esta ha sido una de las decisiones más difíciles que hay que resolver antes de construir uno de estos algoritmos: si se eligen estructuras representacionales complejas, los algoritmos que las utilicen también serán complejos, con lo que la velocidad del sistema será lenta. Si por otro lado se eligen estructuras representacionales simples, es decir, se utilizan características o propiedades de los casos que sean simples, los algoritmos serán mucho más sencillos y por lo tanto más rápidos. Las cuestiones a responder son ¿hasta qué punto es bueno perder calidad en los enlaces en pro de una mayor velocidad? o ¿en qué medida mejora la calidad de la recuperación el aumento de complejidad de las estructuras representacionales?. Hammond [HAM89] completa la lista de cuestiones que deben resolverse antes de implementar cualquier método de indexación o de recuperación:

¿Depende el algoritmo de algún tipo especial de característica?
¿Se puede generalizar el algoritmo sobre distintos dominios? ¿y sobre tareas?
¿Condiciona el algoritmo el hardware del sistema?
¿Depende la eficiencia del algoritmo del número de casos de la memoria?
¿Es el algoritmo cognitivamente plausible?

Kolodner [KOL89] responde algunas de estas preguntas al definir el conjunto de restricciones que debe cumplir todo algoritmo de recuperación:

1. Los mejores casos que verifiquen la prueba deben recuperarse utilizando un conjunto de pistas de recuperación que proporcionen una descripción parcial del término a recuperar.
2. La memoria debe devolver un número de casos pequeño más que uno grande. Si un gran número de casos verifican una descripción pobre, entonces la memoria devolverá o un prototipo, o una generalización, o una petición de más información.
3. La recuperación debe ser rápida. Se hace en el contexto de razonamiento y queremos que el razonamiento sea rápido. Por lo tanto es preferible hacer el trabajo duro durante

la actualización de memoria más que durante la recuperación.

4. El tiempo de recuperación no debe aumentar con el tamaño de la memoria.
5. Deben ser igualmente accesibles generalizaciones y casos.

A continuación se comentarán algunas de las técnicas que se han desarrollado hasta la fecha para los sistemas basados en casos.

2.1 Redes de Discriminación

Las redes de discriminación han sido una de las primeras técnicas y de las que más éxito han tenido durante las primeras épocas de los CBR. Lehnert [LEH87] las introdujo en 1987 y ha sido ampliamente utilizada y mejorada por distintos autores. La siguiente descripción es tal como aparece en el artículo de Rubi & Kibler [RUB88]. La estructura es la típica representación en espacios de estados donde se dispone de un estado de comienzo, unos operadores y un estado objetivo. Lo nuevo es una base de casos que restringe el proceso de resolución de problemas.

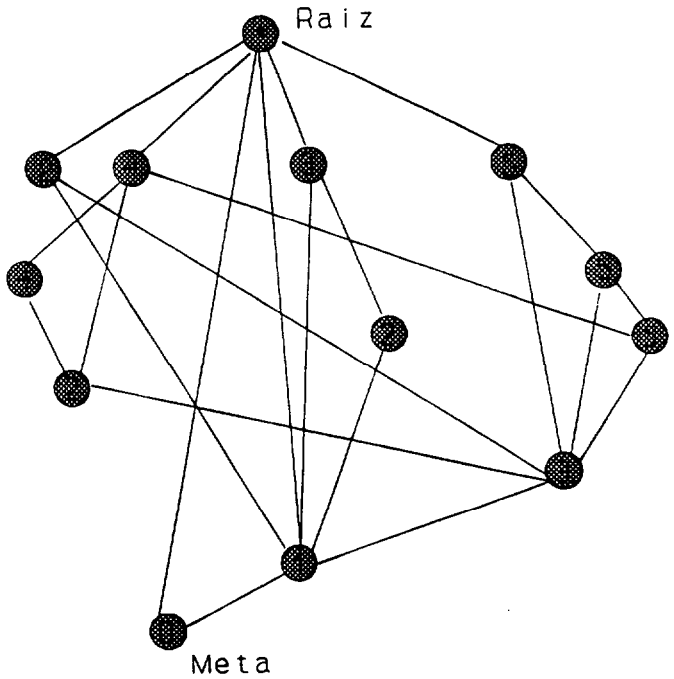


Figura 3: Red de Discriminación

Podríamos considerar la memoria de casos como un conjunto de trazas de soluciones. A

cada estado se le asocia un índice (coarse index). Decimos que dos estados *coinciden en índice (index match)* si tienen el mismo índice. Un camino P para un estado S es la secuencia de estados desde el estado de comienzo hasta S. El camino S coincide en índice con un caso de memoria C si la correspondiente secuencia de índices es idéntica a una subsecuencia contigua de C. El índice puede direccionar múltiples casos en memoria con el mismo índice (número), proporcionando por lo tanto un nivel de abstracción y de generalización sobre las secuencias de soluciones de memoria.

Con estas definiciones ya podemos describir de una manera bastante sencilla el proceso de resolución de problemas:

Hacer una búsqueda normal primero en profundidad pero cancelar cualquier camino P que no coincida en índice con alguno de los casos en memoria.

Este tipo de resolución de problemas puede combinarse con cualquier técnica de búsqueda como Breadth First Search, A*, o Best First Search. No existe garantía de que esta técnica pueda resolver un problema que no esté en la base de casos. Para evitar este problema se incluyó el concepto de soluciones *near-misses*. Un problema tiene una solución *near-miss* si puede ser resuelto con una búsqueda de N-familia. Decimos que un camino P coincide en índice con un caso C si todos menos los primeros i estados de P coinciden en índice con el caso C. En una búsqueda de N-familia se busca iterativamente una comparación positiva de i-índices, con la i variando desde 0 hasta N. Se define la N-familia de un estado como aquellos estados alcanzables con N movimientos o menos.

Bradtke [BRAD88] es otro de los que utiliza las redes de discriminación para resolver el problema del 8-puzzle. Utiliza información del dominio de la aplicación para codificar distintas funciones de índices. La recuperación se realiza enmascarando el estado actual (la primera vez será el estado inicial) dentro de la red de discriminación, podando aquellas ramas que no se ajusten. También en esta ocasión se vale de soluciones near-misses y far-misses para ayudarse a recorrer el árbol en el caso de que llegue a un punto en el que tenga que retroceder.

Sin embargo, no sólo dificulta el funcionamiento de este algoritmo el que el conocimiento sea dinámico, también está el hecho de que las características que se utilizan para indexarlas son en sí mismas muy complejas. El resultado es que los sistemas se apoyan en un ajuste muy fino de los enlaces entre variables para asegurar la identificación de las estructuras de conocimiento adecuadas. El ejemplo tipo lo constituye BATTLE PLANNER de Goodman [GOO89]. Este se basa en una asignación de pesos a todos los campos que se iban a utilizar para la evaluación de la correspondencia, en conjunción con unas jerarquías simbólicas para determinar la distancia entre los valores de los campos simbólicos. Tras múltiples intentos de

ajustar los pesos sobre las características manualmente, el autor reconoce la debilidad de este método, optando por un método automático de inducción (el ID3).

Más recientemente, Sycara [SYC93] ha optado por los árboles de discriminación en combinación con un algoritmo de aprendizaje que se apoya en una jerarquía de episodios generalizados

2.2 DMAP / Concept Refinement Search

El sistema DMAP desarrollado por Martin [MAR89], intenta abordar el problema de las características complejas. Estas son *composicionales*, es decir, están compuestas de múltiples elementos (como son las relaciones causales, o los descriptores del estado actual de un plan). Comprobar restricciones estructurales como estas sería muy costoso (por no decir imposible) mediante redes de discriminación. Sin embargo, no lo sería tanto si se pudieran indexar estáticamente las características con restricciones estructurales a estructuras de memoria. Pero esto plantea problemas debido a que es imposible predecir (y por lo tanto conocer donde indexar) cuales van a ser las estructuras de memoria que se van a necesitar. Se trabaja con asignaciones a variables y no con estructuras estáticas de memoria. Para generar nuevas estructuras de conocimiento hay que determinar cuales de las estructuras de conocimiento existentes contienen la abstracción más específica del nuevo concepto. Saber donde colgar el nuevo concepto (nuevas estructuras como las relaciones causales) permite la herencia de información de indexación de dichas abstracciones.

DMAP, al igual que la aproximación desarrollada por Burke [BUR89], no construye una representación del significado de la historia que está procesando. En su lugar almacena las similaridades y diferencias entre la historias de entrada y las de memoria. En DMAP el programa comienza con unas pocas expectativas de los tipos de historias que va a leer. Su tarea es la de encontrar los nodos de memoria más específicos que satisfacen dichas expectativas y se ajustan a la entrada. Burke sólo considera expectativas específicas (historias) y esperanzas de satisfacer alguna de ellas. [1]

Otro de los autores que han basado sus trabajos en el DMAP es Kolodner con su Concept Refinement Search. Kolodner en [KOL88] genera categorías de memoria (MOPs) organizadas

[1] Una variante similar la constituye el proyecto LABYRINTH [THO89]. En este caso, se utiliza una función de evaluación probabilística en cada nivel de la jerarquía para determinar el concepto que mejor se ajusta al caso actual, y entonces procede al siguiente nivel. El método devuelve el nombre del concepto ***más específico*** que verifica el caso. Normalmente es un nodo hoja (que se corresponde con un caso) aunque si la situación es suficientemente nueva puede ser un concepto no terminal (una abstracción).

en jerarquías de abstracción y en jerarquías de empaquetamiento. Los MOPs (memory organization packets) codifican distintos tipos de situaciones (conocimiento procedural) a la vez que organizan estructuras de indexación. Los índices asociados con cada categoría diferencian entre los términos en la categoría. Si muchos términos comparten el mismo conjunto de índices, se crea un subMOP que los agrupe. Los items (pistas) se organizan con múltiples redes de discriminación redundantes.

En la búsqueda de refinamiento de conceptos, un concepto no es activado hasta que su padre en la jerarquía haya sido accedido, y se haya especificado alguna característica que lo especialice con respecto a su padre. DMAP implementa el refinamiento de conceptos de otra manera. Se puede activar un item en la jerarquía si alguno de sus antecedentes en la jerarquía de abstracción es activado por la prueba (conjunto de características a buscar), si dicha abstracción predice otro concepto, y si alguna especialización del concepto predicho es especificada en la prueba. DMAP tiene la ventaja de no requerir un esquema de indexación redundante. Las predicciones que realiza DMAP son lingüísticas. En el algoritmo de refinamiento de conceptos se han generalizado para buscar eventos en una memoria conceptual.

Visto con un poco más de detalle, el algoritmo tendría cuatro pasos:

1. Inicialmente se parte del conjunto de características a buscar (una prueba). Cada pista o característica de la prueba se transmite a memoria mediante un proceso en serie, y cada una se propaga por toda la memoria en paralelo. La memoria se activa como sigue:
   a) Si la prueba nombra un concepto (ej. (? es.un x)), entonces éste se activa.
   b) Si la prueba es descriptiva (ej. (?, nombre-propiedad, valor-propiedad)) entonces se activan todos los términos que la verifiquen.
2. Como en DMAP, cada nodo activado envía mensajes a las cosas que predice: eventos predicen su secuencia de eventos (sus partes constitutivas), situaciones predicen sus opciones. Al predecir las partes de un evento estamos prediciendo sus características concretas. Un mensaje de predicción tiene 3 partes: fuente, destino, y la relación entre ellas.
3. Como en DMAP, cada nodo activado envía un mensaje, el cual contiene el nombre de la fuente, a cada uno de sus antecedentes en la jerarquía de abstracción para que también se activen.
4. Cuando se encuentran las predicciones y las activaciones se produce el ***'refinamiento de conceptos'***. El concepto que envió la predicción (paso 2) se especializa hasta el nivel de detalle del concepto que envió la activación (paso 3). Los nodos que cooperan para que se produzca esta coincidencia recibirán una atención especial

El resultado es que se han activado varias constelaciones de nodos en memoria. Las constelaciones son casos o conjuntos de casos que verifican la prueba de recuperación. En el caso de que se activen demasiados casos, el sistema devolverá una generalización de ellos, creando dicha generalización si no existiera y fuera posible crearla. Si no lo fuera, el sistema pediría más información.

Esto algoritmos trabajan sobre casos distribuidos jerárquicamente con distintos niveles de abstracción. Donde en cada nodo se incluye información tanto estructural como causal. Esto conduce a algoritmos muy complejos, y que deben ser paralelizados para que su rendimiento pueda ser asumible para bases de casos muy grandes.

2.3 Semántica

Son muy pocos los autores que han intentado como Kolodner incluir la semántica de los conceptos en el algoritmo de recuperación. La mayoría de los sistemas como los ya comentados se contentan con encontrar una similaridad sintáctica entre las descripciones de las situaciones nuevas y las de la memoria de casos. Para ello existen algoritmos que permiten una cierta flexibilidad o una relajación en la calidad de la comparación de las características. Por ejemplo, la herramienta de construcción de sistemas CBR posee este tipo de política. Entre las principales razones para no incluir información semántica en los algoritmos de recuperación se encuentran por un lado, el evidente incremento en la complejidad y velocidad de los algoritmos, y, por otro, los hace mucho más dependientes del conocimiento específico de la aplicación en desarrollo.

Jusisica [JUR94] define el contexto mediante un conjunto de valores que pueden coexistir con el valor o valores que constituyen la sintaxis de la recuperación. El inconveniente de esta aproximación es que es el usuario el que debe especificar el contexto en el que realiza el query (al menos en principio). Este contexto puede ir variando a medida que el proceso de recuperación avanza.

Wall [WALL88] utiliza dimensiones para indexar los casos a partir de conceptos que describen situaciones. Se recuperan aquellos casos conectados a conceptos que son *semánticamente* similares al concepto de la situación actual. Para favorecer la recuperación orientada a metas incluye una descripción semántica de las metas en el concepto de situación que está clasificando. A cambio, deben especificarse *explícitamente* los conceptos y las relaciones entre ellos. Con esto se consigue que el ingeniero del conocimiento pueda tener una justificación taxonómica a la hora de la recuperación. La recuperación lo que intenta es clasificar la estructura del testigo, y se hace en función de la correspondencia con una

estructura de grafos (en lugar de a una lista de tokens). Esta estructura de grafos representa el fondo semántico del ajuste del token. De todo esto se deduce que un matching con esta metodología implica una similaridad semántica. Rajendra considera que el conocimiento sobre el medio específico (en este caso la toma de decisión militar) es crucial a la hora de seleccionar un plan.

Otro de los autores que considera la existencia de una serie de características del dominio que restringen la elección de una estructura y una técnica de indexación, es Kopeikina [KOP88]. En este caso dichas características no son de origen semántico, sino más bien estructural. La más importante es que el dominio (control del tráfico en tiempo real) no soporta un conjunto único de atributos que sean siempre los discriminantes entre situaciones.[2] Este hecho impone el que no se pueda utilizar un árbol jerárquico de discriminación. En su lugar se utiliza una 'fábrica de indexación', que representa un mapa conceptual de las situaciones que el sistema debe afrontar. Esto difiere de las redes convencionales de discriminación en dos puntos:

1. La estructura no es de árbol sino de grafo con enlaces laterales.
2. Las decisiones de discriminación no tienen que ser únicas.

El indexador es la parte del sistema que recupera casos que son similares a la situación actual. Dada una situación y un punto de partida, el indexador produce una lista de casos almacenados según una estimación de su relevancia.

La fábrica de indexación consiste en una red con tres tipos de nodos:

1. La estación de índices: que representa una caracterización de una situación.
2. Nodos de envío (dispatcher): conecta estaciones de índices. Son similares a las sentencias 'case' de un lenguaje de programación. Interpretan los criterios del dominio para discriminar entre las caracterizaciones representadas por las estaciones de índices que conecta.
3. Puertos de envío: etiqueta los enlaces entre los nodos de envío y las estaciones de índices. Se correspondería con cada una de las opciones de una sentencia 'case'.

La búsqueda sería como sigue: el objetivo es el de encontrar una buena caracterización de la situación actual. Primero se lanza un peso de una estación de índice a otra. El peso indica la confianza de que la estación actual es buena. Según se pasa el peso, este se divide o se reunifica. Llegado un tiempo máximo *t* o la falta de un determinado dato, el peso se para. La

[2] Ello se debe a otras restricciones que impone la aplicación como son: procesamiento restringido en el tiempo, y la necesidad de mantener y representar casos que evolucionan a lo largo del tiempo.

mejor caracterización es aquella para la que el peso fue máximo. ¿Qué sucede si la red es suficientemente grande? que el rendimiento del sistema decrece demasiado, al tener que aumentar el tiempo límite t para que de tiempo a recorrer el grafo.

2.4 Sistemas Dinámicos

La naturaleza dinámica de estos sistemas es evidente. Según el sistema evoluciona, la base de casos crece, y la complejidad de la memoria puede hacer que los algoritmos de recuperación se vuelvan ineficaces. Por ello, y tal como advierte Schank en [SCH82], el sistema debe ir adaptándose a medida que recibe nueva información. Adaptarse no sólo significa que debe asimilar nueva información, sino que también debe evaluar la validez de la que ya tenía en memoria. 'Índices que eran útiles dejaran de serlo porque las instancias únicas que direccionaban ya no lo son'.

Owens [OWE89] afirma que si el conocimiento fuera estático y se pudieran examinar las características en un orden determinado, tendríamos un árbol de decisión extremadamente simple. Los índices deben ser lo suficientemente abstractos para que la solución indexada sea aplicable a otros casos que comparten dichas características abstractas. Pero también tienen que ser lo suficientemente ambiguos para que los casos de un dominio puedan aplicarse a problemas en otro dominio, asumiendo que comparten algún contenido temático abstracto. También tienen que ser lo suficientemente flexibles como para permitir una ampliación del vocabulario a medida que se lo presentan nuevas situaciones al sistema. El algoritmo de Owens actualiza constantemente el conjunto de características que tienen más contenido de información. El sistema reorganiza dinámicamente el árbol con lo candidatos que le quedan en cada paso (recuperación incremental).

Sycara y Navinchandra [SYC89] desarrollaron otro de los sistemas que adaptan dinámicamente el conjunto de índices. Estos se generan cuando se produce alguna de estas situaciones:

- Se dispone de nueva información.
- Se violan expectativas.
- Se utilizan relaciones causales para encontrar submetas que ayuden a completar el objetivo

De la última situación se deduce que la adaptación de índices es una característica innata de aquellos sistemas que, como el de Martin anteriormente comentado, utilizan características composicionales que 'obligan' a que se creen nuevos conceptos y relaciones a medida que

el sistema evoluciona. Sin embargo, para que la reindexación funcione correctamente, el dominio debe estar sólidamente modelizado mediante un modelo causal como en los ejemplos anteriores.

También Fox [FOX94b] ha desarrollado un sistema que reajusta dinámicamente sus índices a medida que se detectan fallos en su funcionamiento. En este caso, cuando se deduce que el sistema no ha recuperado el mejor caso, se construye un nuevo índice que permitirá distinguir al verdadero caso del recuperado erróneamente (de la misma forma funciona el sistema CHEF de Hammond [HAM89]). El problema está en como detectar que se ha recuperado un caso que no es el óptimo. Fox y Leake se apoyan en un modelo jerárquico de funcionamiento del sistema, el que realmente es un sistema model-based, que coopera con el sistema CBR. Como sucede con todos los sistemas model-based, para que su arquitectura funcione correctamente, es necesaria la codificación tanto de lo que se llamaría un 'buen comportamiento' como de todos los posibles 'modelos de fallos'.

2.5 Indexación versus no indexación

A partir del desarrollo de algoritmos paralelos se ha generado una polémica acerca de si realmente son útiles los índices en una memoria de casos. Por un lado están los autores niegan la utilidad de tal esquema, y por otro están los que no sólo lo defienden sino que llegan a definir distintos tipos de índices según sea el uso que se quiera hacer de él.

Pazzani [PAZ89] considera que con un conjunto de índices más una información adicional del tipo de metas se consigue que la recuperación sea más fácil a la vez que un caso puede ser útil en múltiples circunstancias. De la misma idea es Rissland [RIS88] que considera que un indexado correcto minimiza la búsqueda.

En contraste con estas opiniones a favor de una especialización de los índices podemos encontrar otras dos que los desechan: las de Thagard [THA89] y Waltz [WALT89].

El primero considera que la recuperación es un proceso altamente paralelo que se apoya en restricciones semánticas, estructurales, y pragmáticas (metas). También expone las razones de su postura:

La indexación no es positiva porque:

Es un proceso serie.
Se apoya demasiado en las metas.
No da importancia a determinadas características semánticas

No se apoya en similaridades estructurales.
Se impide la recuperación de analogías menos relevantes.
Precisa demasiado preprocesamiento.

Lo que propone es un proceso masivamente paralelo mediante el que se buscan razones semánticas, estructurales, y pragmáticas que permitan recuperar una analogía para resolver un problema, dar una explicación, alcanzar una solución....

Waltz considera que si la recuperación es un proceso paralelo, los índices no son necesarios computacionalmente, a la vez que con la utilización de índices sólo se contemplan características superficiales. Justifica que la indexación no juega casi nunca un papel importante, como cuando hay que seleccionar una acción, diagnosticar, o valorar una situación. La recuperación es viable porque se realiza en paralelo. Cada caso se almacena en memoria activa, intentándose una comparación exitosa de cada caso con la situación actual. La cuestión de la selección de un caso es permitir recuperar características *profundas* de evaluación *rápidamente* dadas sólo unas determinadas características *superficiales* accesibles.

3 Vocabulario de indexación

Con el vocabulario de indexación se quiere hacer referencia al conjunto de características que deben utilizarse en la recuperación de los casos. De la elección del conjunto correcto de estas características depende la recuperación del caso adecuado, y por lo tanto la calidad del sistema CBR.

Al menos en principio, este conjunto debería estar formado por todas las características utilizadas para representar los casos en memoria. En la práctica, algunos subconjuntos del vocabulario representacional completo son más útiles que otros en determinadas situaciones, ya sea porque son generalmente más fáciles de computar --en el caso límite se proporcionan directamente en la entrada-- o porque es más probable que aparezcan en la recuperación de un caso de memoria. Desafortunadamente, este tipo de información acerca de los costes y ventajas de utilizar diferentes conjuntos de características en diferentes circunstancias puede no estar disponible.

En definitiva, el principal problema que hay que afrontar está en decidir si utilizar aquellas características que son fáciles de computar, o quizás sólo las que aparecen en la descripción inicial de la situación del problema (características de bajo nivel -- low level), o si debería dedicarse algún esfuerzo a la derivación de propiedades más abstractas que las meramente dadas en la descripción inicial del problema. Esta última aproximación depende básicamente de tener un conocimiento previo de los tipos de características abstractas que es posible derivar en la recuperación de casos similares. Por lo tanto, ahora surge la cuestión de si realmente existen tales características y, siendo así, si podemos descubrir lo que son. Si se pueden responder a estas preguntas afirmativamente, entonces habrá que llegar a un convenio entre las dos aproximaciones anteriores. La primera de ellas dedica un mayor esfuerzo al 'back end' del sistema basado en casos, es decir, a la valoración de lo apropiado que es el caso recuperado respecto a la situación actual, y adaptarlo para que reúna las características actuales. En contraste, la segunda aproximación pone un mayor énfasis en el 'up front', en el análisis del problema.

Pero no sólo se trata de decidir si utiliza características de alto nivel o de bajo nivel. El estudio ralizado por Seifert [SEI94] junto con psicólogos de la universidad de Michigan demuestra que las características realmente útiles (predictivas) pueden ser tanto de bajo nivel como de alto nivel y que, para poder identificar cuales son estas características predictivas, es necesario establecer las causalidades inherentes en ellas mismas.

Además de estas cuestiones generales, existe la tarea de desarrollar vocabularios

particulares útiles en determinados dominios y tareas. Otra de las cuestiones es hasta qué punto se puede automatizar este proceso.

Otras consideraciones importantes a considerar son:

1. ¿Cuales son los descriptores que serán utilizados como índices y de qué elementos del vocabulario se componen?
2. ¿Cuales son los descriptores asociados a cada situación?
3. ¿Cual es la estructura de las etiquetas?
4. ¿Como se organizan las etiquetas?
5. ¿Qué principios funcionales pueden aplicarse en el desarrollo de los vocabularios? ¿Puede servir como base una teoría de utilidad?

3.1 Características de Bajo Nivel

Comencemos con el modelo más simple de todos: utilizando únicamente un conjunto de características de bajo nivel con las que realizar las comparaciones. Son pocos los trabajos que cumplen esta característica. Uno de los más sencillos es la herramienta de construcción de sistemas basados en casos CBR- Express, la cual utiliza una serie de características de bajo nivel para guiar la recuperación de los casos más similares. El sistema no es capaz de modificar la respuesta del caso recuperado para adaptarlo a la situación actual. La codificación de las características se realiza en base a una batería de preguntas asociadas con cada uno de los casos de memoria. Los tipos de respuestas son los normales a cualquier lenguaje de representación (por ejemplo un número real, una lista de valores simbólicos permitidos....). La selección de las preguntas asociadas a cada caso y el orden de prioridad a la hora de hacerlas al usuario vienen definidas por el diseñador del sistema.

El sistema OGRE [DOH89] implementa otro sistema independiente del dominio a partir de un conjunto de características de forma similar a la herramienta anterior. Los factores se definen mediante descriptores que contienen la siguiente información:

- Nombre del factor, ej. Temperatura
- Tipo de dato, ej. número real
- Valores legales permitidos para el dominio en particular, ej. entre -300 y 200
- Valor por defecto
- Método por el que se comparan dos valores de este factor, ej. que *T1* esté entre 10 grados de *T2*
- La importancia relativa respecto a otros factores del dominio

- Método para calcular el valor de este factor en función de los valores de otros factores.

Un sistema ligeramente más complejo es BATTLE PLANNING [GOO89]. Este complementa la lista de características con una jerarquía organizacional que le permite realizar generalizaciones sobre información puntual. Dichas generalizaciones permitirán realizar la recuperación de un mayor número de casos. Sin embargo, tal como reconoce Goodman, es preciso introducir información causal en la construcción de los índices. La información causal es la que va a permitir aprender, ya que posibilita la construcción de nuevos índices que expliquen las situaciones nuevas.

SWALE [KAS89], sin llegar a implementar una red causal pura, construye una red de conceptos basada en la Teoría de Dependencias Conceptuales de Schank. Como se comentó anteriormente, un caso se representa mediante una Explanation Facility (XPs), las cuales son ejemplos paradigmáticos de situaciones estereotípicas. Lo que se intenta es construir explicaciones a partir de los casos almacenados y adaptarlas a la nueva situación. Las características se enlazan mediante nodos que representan el grado en el que están relacionadas. Estas relaciones causales explícitas facilitan la adaptación de casos. El modelo de proceso de SWALE es muy similar al de CASEY [KOT88]. CASEY también utiliza un conjunto de características combinadas con un modelo causal y un sistema de explicaciones causales. Este modelo combina el conjunto de factores en *estados* que caracterizan la situación del paciente. Los distintos estados se relacionan entre sí mediante enlaces probabilísticos, estableciendo relaciones causales entre ellos. Sin embargo, la principal diferencia entre SWALE y CASEY radica en que las explicaciones de SWALE son muy abstractas y necesitan una gran cantidad del conocimiento del dominio para evaluar y reducir las diferencias. Las explicaciones de CASEY son muy concretas, y para evaluar o reparar una diferencia sólo necesita examinar información local sobre los estados y las evidencias.

El último sistema a comentar en esta sección es CICLOPS [NAV88]. Este unifica las aproximaciones anteriores en un sistema que utiliza una ajuste directo, un ajuste flexible, y otro basado en la sistematización. El primero realiza una comparación directa entre los atributos del caso base y el objetivo, tomando en consideración el caso base en el caso de que la comparación tenga éxito. El ajuste flexible confronta los atributos del caso base y objetivo desplazándose por una jerarquía de objetos predefinida, tomando en cuenta aquellos casos que sean cubiertos por los niveles altos de la jerarquía. El último tipo de ajuste establece que, para poder encontrar un ajuste analógico entre el caso base y el objetivo es más importante encontrar relaciones causales comunes entre los atributos de dichos casos, que sólo entre atributos comunes. El ajuste analógico se consigue en CICLOPS almacenando, junto con cada caso, una explicación causal de las metas y submetas del caso.

En general, el proceso analógico que se consigue únicamente con características de bajo nivel es muy limitado, ya que existen muchas relaciones que favorecen el proceso analógico que sólo se pueden identificar en niveles abstractos de modelización. La utilización de conocimiento causal, sin permitir a la vez una abstracción de la información, no mejora en exeso esta cuestión ya que, al estar sujeto el modelo causal a la lógica de primer orden, se hace muy laborioso constuir abstracciones de datos.

3.2 Características de Alto Nivel

La otra gran opción es la de utilizar, en la recuperación, la información en principio no incluida en la descripción del caso a estudiar. Estas abstracciones permiten la recuperación de casos *fuera de contexto*, que ampliarán el abanico de experiencias sobre el que trabajar. Esta corriente tuvo su origen en el trabajo que sobre memoria dinámica realizó Schank en 1982 y que tan recurrido es en toda la bibliografía de CBR. Schank intentaba explicar porqué la gente asimila una situación pasada con una nueva, a menudo no relacionada. Propuso que tal rememorización ocurría para ayudar a formar una respuesta adecuada a la nueva situación, y que dichas situaciones previas se almacenaban como episodios indexados por una estructura de generalizaciones compleja. Durante el proceso de comprensión, se asimila cada parte en términos de dichas generalizaciones. Dichas generalizaciones indexan componentes de otros casos, posiblemente fuera de contexto, y el resto del proceso también tiene acceso a dichos casos, de forma que están disponibles para ayudar en la construcción de una respuesta adecuada al nuevo caso. Son necesarias dos características de memoria dinámica para poder recuperar un caso fuera de contexto: generalizaciones parciales del caso y representaciones episódicas del caso. Primero, una generalización del caso no es una generalización del caso entero, sino sólo de parte del mismo; se asimilan diferentes partes del caso en términos de diferentes generalizaciones. Segundo, la memoria de un caso debe ser episódica; debe contener un conocimiento detallado de la secuencia de eventos que ocurrieron en el caso, en lugar de contener una estructura única y estática que resuma el caso. Sin una generalización parcial del caso, el acceso a un caso fuera de contexto requeriría un alto nivel de generalidad inútil en el que casi cualquier otro caso sería considerado como relevante. Sin la representación episódica del caso no sería posible la generalización parcial del caso. Estas dos características juntas son las que permiten la recuperación de casos fuera de contexto.

En definitiva, como describe Birnbaum [BIR89], lo que Schank defiende es que se pueden encontrar casos que comparten características abstractas y no compartir ninguna de las características de bajo nivel. Esta facultad nos permitiría aprender lecciones en un dominio y aplicarlo en otro que superficialmente no tiene casi nada o nada en común. Las estructuras que permiten implementar este conocimiento se denominan (siguiendo la nomenclatura de

Schank) *Thematic Organization Points (TOPs)*, las cuales representan patrones de planes e interacciones de metas abstraídas a partir de planes específicos y las metas que intervienen en ellos. En otro trabajo [BIR88], Birnbaum denomina este tipo de conocimiento *conocimiento estratégico*. Por ejemplo, consideremos el concepto de estrategia en una competición; una situación en la que un agente adquiere la oportunidad, o de ganar la competición o pierde todas sus posibilidades en favor de su oponente. O consideremos el concepto de *sacrificio*, en el que el agente permite que el oponente alcance una meta para que él gane una meta más importante. Ambos conceptos son significativos en cualquier situación competitiva, sin importar el dominio en el que se desenvuelva la competición. Por lo tanto, estos conceptos deben representarse en términos de un vocabulario abstracto aplicable a cualquier tipo de planificación competitiva o dominio similar. Birnbaum afirma que la adquisición de nuevos conceptos estratégicos mejoraría significativamente su habilidad de recuperar casos relevantes. El aprendizaje de un nuevo concepto estratégico daría lugar a un mejor rendimiento en diferentes dominios. Propone el desarrollo de métodos de aprendizaje basado en las explicaciones (EBL) capaces de derivar tales tipos de conceptos a partir del análisis de casos específicos (posteriormente se comentarán los sistemas EBL en relación con el vocabulario de indexación). Sin embargo, es preciso recordar que una aplicación específica también incluye conocimiento específico del dominio, como la 'física del dominio' - ej. reglas de juego-, el conocimiento acerca de las metas que caracterizan el dominio, o planes específicos para alcanzar dichas metas.

Otro de los autores que considera que para que un caso tenga una aplicabilidad considerable es necesario que las características con las que es indexado reflejen alguna abstracción temática, es Owens [OWE88]. Además, el análisis necesario para derivar esta abstracción temática debe hacerse en el momento de adquirir el caso. Sin embargo, esto condiciona de alguna manera la recuperación. Los mismos tipos de índices de abstracción que proporcionen una buena rememorización a través de dominios distintos son probablemente los más difíciles de detectar en los casos de entrada, haciendo que el primer paso de los algoritmos de CBR, o preanálisis, sea mucho más difícil. Como deja notar Owens en [OWE89], la transformación de los datos de bajo nivel de entrada en dichos descriptores va más allá de la simple asignación de pesos de importancia a las características o la adición de nuevas características que son una combinación booleana de algún conjunto existente de las mismas. Extraer dichos descriptores abstractos puede requerir de la asignación de variables y de reglas de inferencia de complejidad variable. Idealmente estos índices deberían satisfacer un conjunto de restricciones:

- Que sean fáciles de extraer a partir de los descriptores de bajo nivel
- Que sean utilizables como pistas de búsqueda y recuperación
- Que categorizen los casos de memoria respecto alguna dimensión interesante, por la que los casos que se indexan de forma similar son los que se relacionan de una

manera funcional. Por ejemplo, si un sistema sabe de muchos casos que describen una cierta clase de fallos del sistema consideran apropiada una determinada estrategia de recuperación, deberían compartir algunos de los enlaces o índices.

- Además, deben ser lo suficientemente abstractos para poder aplicarlos en otros casos y lo suficientemente flexibles como para poder aplicarlos en otros dominios.

Como se puede ver, estas restricciones son conflictivas en algún sentido. Las características más fáciles de extraer no son necesariamente las más interesantes funcionalmente, ni son las más fáciles de utilizar como pistas de recuperación.

Cuando se habla de *similaridad temática,* es decir, de los criterios utilizados para considerar que un caso es relevante, se hace referencia a las relaciones causales abstractas, como utiliza Leake [LEA92], y a las orientadas a metas que se mantienen entre los elementos del caso. Martin [MAR89] sostiene que, mientras que dos dominios puede que no tengan relación aparente a nivel de características de bajo nivel, puede haberla a nivel temático. El vocabulario de este nivel temático es una expresión de las conexiones causales y las metas de planificación de los actores implicados. Las estructuras temáticas de los casos implicados comparten el mismo vocabulario causal, pero no tienen que poseer la misma estructura causal.

Koton [KOT89] advierte otra ventaja que se consigue implementando descriptores abstractos: no es necesario encontrar un caso que se ajuste completamente al nuevo caso. Los descriptores abstractos consiguen agrupar múltiples condiciones en una sola restricción, reduciendo la complejidad de la evaluación de la correspondencia. El problema así planteado (con descriptores abstractos únicamente) es descompuesto *posteriormente* en subproblemas independientes. Estos módulos son los que se presentan en memoria para su comparación. Se activan los módulos almacenados que son similares a la descripción abstracta. A partir de las aportaciones de los distintos módulos se construye la nueva solución para todo el problema.

Una postura intermedia pragmática y oportunista la mantiene Hammond en [HAM91]. Considera que la elección de características de bajo nivel o de características abstractas depende de la naturaleza de la tarea a resolver. La estructura de la situación debe ser explícitamente especificada facilitando que sea utilizable como índice de almacenamiento y de recuperación. Sin embargo, existe una limitación a este principio. No se deben indexar metas por características que sean extremadamente comunes y que estén asociadas con muchas otras metas. En definitiva, cuando se precisen reacciones rápidas de la base de casos se deben utilizar características de bajo nivel. Para análisis profundos, características abstractas.

Compartimos con Hammond que la posición más prudente consiste en tomar una posición

intermedia. Apoyarse demasiado en abstracciones puede conducir a la recuperación de demasiados casos; la postura contraria a no recuperar ninguno.

3.3 Principios Funcionales

Cuando se desarrolla un vocabulario de indexación, ¿En qué se basa la elección de las características? ¿Existen principios funcionales que orienten la construcción de dichos descriptores?

Pazzani [PAZ89] distingue dos tipos de índices: los predictivos y los explicativos. Un índice predictivo se utiliza para encontrar un esquema de datos que describa el resultado de un evento determinado. Un índice explicativo se utiliza para encontrar un esquema de datos que describa la causa (la justificación) de un evento determinado. Las características predictivas son aquellas características de una generalización que son únicas a dicha generalización. Con estos tipos de índices más una información adicional del tipo de metas que es capaz de resolver se consigue que la recuperación sea más fácil a la vez que un caso puede ser útil en múltiples circunstancias.

Ashley [ASH89] piensa que no basta con definir una estructura de organización de los índices más o menos eficiente, sino que hay que saber utilizarlos y aprovechar la información que direccionan. El índice del caso debe reflejar las similaridades y las diferencias importantes entre la situación del problema y los casos pasados. Así, en función de estas similaridades / diferencias define dos tipos de índices:

1. Índices de Generalización y de Abstracción: en los que las similaridades y las diferencias implican la descripción de los problemas y de los casos, y donde los índices se derivan de jerarquías de abstracción empleados en el modelo analítico.[3]
   Aquí distingue entre tres tipos más de índices:
   a) Factores (dimensiones) presentes en el caso. Los factores son el conjunto de hechos que favorecen o no una determinada salida. Los casos pueden ser ejemplos positivos o negativos de la salida.
   b) Relaciones causales 'elegidas' de entre las presentes en el caso. Los hechos y los casos pueden estar ligados al resultado del caso vía una cadena de inferencia.
   c) Conceptos de los que el caso es un ejemplo positivo o negativo. (ej. predicados de una regla utilizada para explicar una solución).

[3] Los modelos analíticos son los modelos de razonamiento que no se apoyan en los casos (como los modelos deductivos basados en reglas)

2. Índices Derivacionales: las similaridades y las diferencias implican las soluciones que se intentaron con técnicas analíticas y de ahí se derivaron los casos y los índices. Son las características de las soluciones intentadas en un caso. Por ejemplo planes de solución, planes fallidos, conflictos entre metas...

En la sección anterior comentamos brevemente la opinión de Owens sobre lo que debe ser un índice. Owens en [OWE89] analiza la forma en la que se podría diseñar el vocabulario aplicado a un sistema de planificación. Si el objetivo de la recuperación de los casos es el de anticipar y evitar fallos, una buena manera de indexar los casos que caracterizan los errores de planificación es a través de las decisiones que llevaron a dichos errores. Se puede utilizar como índice en la recuperación de casos una decisión particular. El planificador puede determiñar a qué conjunto pertenece la situación actual, y consecuentemente en qué sentido debe tomarse una decisión, mediante la comparación de aquellos casos en los que la decisión derivó en una salida aceptable con los que condujo a una situación desfavorable. Al utilizar puntos de decisión como índices, un sistema puede extraer descriptores que sólo tienen validez heurística. Es decir, debe desarrollarse la heurística necesaria para transformar dichas metas en índices que sean utilizables en la recuperación de casos. La condición puede resultar en una interacción inesperada de múltiples decisiones aparentemente independientes. El sistema puede que no reconozca que el descriptor es aplicable en aquellos casos en los que las decisiones no posean el grado de generalidad suficiente. En [OWE93] es más explícito y considera que la selección de las características vendría dada por su relevancia funcional. Asi, para un sistema de planificación existirían cuatro tipos de relevancia funcional.

1. Failure related: que se correponden con características con la misma causalidad
2. Symptomatic: cuando existe una correlación estadística, pero no una relación causal explícita
3. Recovery related: que vendrían en función del estado actual del sistema, no de la situación actual. (ej. una meta)
4. Direct: se corresponden con características causales pero que no pertenecen a ningún patrón de representación. Están ligadas más por el objetivo que por la función que realizan dentro del sistema CBR, no del dominio.

El problema también podría plantearse en términos de qué características utilizar en la recuperación sin tener que comprobar cada posibilidad. O dicho de otra forma, ¿qué aspectos de la entrada utilizar? ¿qué estructuras internas del sistema son importantes para procesar una experiencia?. Como apriori no es posible conocer todos los casos que se pueden presentar al sistema, este deberá decidir dinámicamente cuando es rentable almacenar un nuevo caso en memoria y cuando es conveniente olvidarlo. La respuesta puede parecer evidente: idealmente, un programa debe almacenar aquellos casos cuyos métodos de adaptación de casos se vayan

a necesitar en el futuro [HUN89]. Pero, por supuesto no es posible anticipar todas las posibles metas futuras, ni identificar con precisión en qué medida será útil un caso en el futuro. Lo que sí se puede hacer es identificar los que son probable que sean útiles en el futuro (*paradigm cases* -PC).

Como no es posible comparar cada experiencia con cada meta potencial de cara a encontrar casos que serán útiles eventualmente, el problema se puede simplificar de tres maneras:

1. No se necesitan tener todas las metas futuras
2. Cuando se encuentra una meta que se puede beneficiar del descubrimiento de un PC, se pueden delimitar los contextos en los que es posible encontrar tal paradigm case.
3. Es posible tomar medidas para limitar el coste de comprobar si una experiencia puede utilizarse como paradigm case.

Aprender de la experiencia lleva trabajo. Un sistema de aprendizaje debe ser capaz de identificar lo que necesita conocer y prepararse para reconocer y procesar dicha información cuando esté disponible. Un programa de adquisición de PC debe ser capaz de elegir que metas se van a beneficiar de dichos PC (mediante la introspección), y que aspectos de las entradas van a indicar la presencia de tales casos. Para poder encontrar la información deseada en la experiencia, un sistema de aprendizaje debe ser capaz de saber que está buscando. Los casos de entrada se pueden convertir en PC para la consecución de una determinada meta importante. Este tipo de aprendizaje es *oportunístico*. La planificación de la adquisición del conocimiento oportunístico facilita el descubrimiento de PC, incluso en contextos diferentes de el del que surge el problema de rendimiento, sin tener que realizar una búsqueda exhaustiva a través de todos los contextos posibles en los que podría ser relevante una experiencia. Cuando un caso demuestra una particular carencia de conocimiento (falla), se utiliza para generar una *meta problemática anticipada* para adquirir conocimiento que prevenga de tal fallo. En definitiva, los casos que merece la pena recordar son aquellos que son probables que faciliten el cumplimiento de una meta problemática anticipada.

Hasta ahora todas las aproximaciones comentadas se pueden incluir dentro de dos tendencias claramente definidas: (acerca de la naturaleza de las características de deben utilizarse en la indexación)

a) Utilizar un conjunto de características de entrada sin procesar (o sólo sintácticamente)
b) Utilizar junto con estas características un vocabulario de las metas e interacciones entre planes.

Sin embargo, para Hammond [HAM89] estas diferencias son artificiales. Es mejor utilizar

un análisis funcional de las metas de la recuperación de memoria y el uso real de los términos que se recuperan. Para seleccionar el conjunto de índices hay que tener en cuenta tanto aquellos que vienen impuestos por las restricciones de la tarea en particular (planificación, ...), como los que provienen de los requisitos de la memoria. Una vez hecho esto, se puede construir una definición de lo que es una definición del vocabulario adecuado para el acceso a memoria:

- Se comienza especificando los requisitos de la tarea y representacionales.
- Entonces de determina el papel que juega la memoria en términos de dicha tarea.
- Finalmente, se hace uso de la representación inicial de la tarea como punto de partida para la construcción de los elementos representacionales para el acceso a memoria.

Lo que se pretende es conseguir una descripción funcional del problema que pueda utilizarse como un conjunto de características para acceder a términos en memoria. Cuando se recupera un caso de memoria, cuando se hace uso de la rememorización, este caso comparte pocas características superficiales con la situación inicial, aunque el ajuste es casi perfecto a un nivel de mayor análisis de la causalidad del problema. También considera Hammond que la rememorización proporciona una valiosa recomendación acerca del problema que se estudia y que es descrito por las características utilizadas (¿metaconocimiento?). También puntualiza que las características que tienen en común la situación inicial y las restantes son características que el planificador computó anteriormente para resolver un problema particular del área en cuestión (en el trabajo de Hammond planificación de recursos). Es por ello que el vocabulario utilizado en la recuperación de memoria debería incluir *todo* lo que puede representar un sistema. No se trata de que una recuperación de memoria utilice sólo descriptores abstractos de relaciones causales, sino que *también* debe incluir dichos descriptores.

En la última aproximación que comentaremos también se considera el problema del vocabulario de indexación como un problema de aprendizaje. Barletta [BARL88] trata de adaptar las técnicas EBL para identificar los índices importantes a partir de un conjunto de características relevantes. Parte del objetivo clásico de los sistemas EBL: crear explicaciones de lo que hace a un caso particular relevante para una descripción particular del problema. Se trata de justificar la existencia de las acciones ejercidas en el caso respecto a las condiciones iniciales. No se trata de una validación, es decir, de explicar el diagnóstico (salida) final del caso. Para ello dispone del conocimiento específico del dominio (que es incompleto), de las acciones de diagnóstico y reparación que pueden realizarse, y algún conocimiento de las relaciones causa-efecto que se pueden dar en el entorno del problema (en el caso de Barletta células robóticas).

Propone que el sistema razone con esta teoría del dominio para dividir los índices potenciales en tres categorías: relevantes seguros, irrelevantes seguros, y posiblemente relevantes (o irrelevantes). Los índices relevantes pueden recomendar o contraindicar un caso.

Para indexar un caso, se extraen los índices irrelevantes; entonces los índices restantes se utilizan para organizar los casos en memoria según el siguiente esquema. Los índices relevantes se organizan en *índices primarios*. Los índices posiblemente relevantes se utilizan como *índices secundarios*. Cuando llega un nuevo problema, los índices primarios se utilizan para determinar un conjunto de casos aplicables, y los secundarios para seleccionar entre dicho conjunto. Los índices secundarios pueden eventualmente promocionarse a índices primarios o pueden desecharse como irrelevantes ya sea porque se completa la teoría del dominio o porque se utilizan técnicas inductivas. El sistema realiza una justificación en lugar de una validación. Esto reduce el conocimiento del dominio necesario para resolver el problema. El sistema sólo necesita suficiente conocimiento acerca del propósito del caso y las relaciones causales en el domino para poder explicar porqué se siguió una determinada secuencia de acciones. Si la meta del sistema fuera la validación, la teoría del dominio debería ser mucho más completa.

3.4 Vocabulario y Explanation Based Learning

En la sección anterior se introdujo el concepto de aprendizaje. Un sistema CBR debe ser dinámico en el sentido de que debe aprender de su experiencia. Cuando los índices suministrados por el diseñador han demostrado ser insuficientes para recuperar algún caso válido de memoria, se hace necesaria una reestructuración del conjunto de índices.

Uno de los trabajos más generales es el de Sycara [SYC89]. Considera que utilizar un conjunto de índices predeterminado limita la utilidad de los sistemas de resolución de problemas, especialmente en situaciones en las que se generan nuevas metas y submetas durante la resolución de problemas que no se corresponden con ninguno de los índices predefinidos. Estas situaciones hacen necesaria una transformación o una generación de índices.

Transformación de Índices

A continuación se comentan algunas técnicas de transformación.

Elaboración de Índices: Consiste en añadir más detalles. Puede elaborarse el estado actual de una situación para que proporcione un mayor nivel de detalle o pueda condensarse en principios genéricos. Añadir más detalles a un índice proporciona información extra que

podría permitir recuperar un caso que de otra forma se perdería.

Abstracción de Índices: Se pueden recuperar casos similares si se abstraen hacia conceptos generalizados. Para especificar abstracciones se utilizarían otros índices de la definición del problema.

Mutación de Índices: Investigadores trabajando en la psicología de la creatividad han encontrado evidencias de que los humanos utilizan mutaciones sintácticas en la resolución de problemas. Algunas heurísticas que están se considerando implican cambios en el tamaño (ej. minimizar), sustituciones (ej. de ingredientes, de fuentes de potencia), alternativas (ej. usos alternativos), inversión (ej. invertir causas y efectos).

Estas transformaciones producen índices que pueden no reconocerse utilizando las características la definición original del problema como índices.

Generación de Índices

Durante la resolución de problemas se pueden generar nuevas especificaciones y metas a medida que se descubren interacciones desconocidas y nuevos requisitos.

Algunos métodos de generación de índices son los siguientes:

Generación de Submetas Basada en la Explicación: Las metas y submetas que se generan dinámicamente pueden utilizarse como índices de memoria. La hipótesis es que aunque un caso no tenga las mismas metas explícitas que la situación actual, sí que puede que compartan las mismas submetas. Se hace preciso la habilidad de buscar en las explicaciones causales de los casos y en las tareas de resolución de problemas. Se podría sustituir una meta por submetas que contribuyen (causan) a la misma. Por ejemplo, si no se supiera eliminar un problema, ¿se podrían eliminar sus causas?.

Simulación Cualitativa: Una simulación ayuda en la identificación de ciertas interacciones entre los subcomponentes, e inconsistencias en el diseño. Las metas para resolver dichos subproblemas se utilizan como índices para recuperar casos relevantes.

Generación dinámica de explicaciones dirigida por intenciones: Siempre se ha supuesto que si un caso es importante para resolver una determinada meta en la situación 'a mano', contendrá una referencia a dicha meta. Esto no siempre es verdad. Puede que un caso no haga una referencia directa a una meta actual. Puede que sea necesario encontrar una analogía entre

el índice y el caso. Así, la analogía consistiría en encontrar una comparación o un ajuste de relaciones en una explicación causal de caso. La idea es generar dinámicamente dichas explicaciones utilizando la meta actual como propósito de la explicación.

Esta clasificación está incompleta en cierta medida. Como puntualiza Barletta en [BARL88], las aproximaciones desarrolladas hasta la fecha se apoyaban en métodos inductivos y en lenguajes de descripción restrictivos para reducir el conjunto de índices y generalizar los índices resultantes. Estas aproximaciones tuvieron un relativo éxito en los primeros sistemas CBR. Sin embargo, se han detectado una serie de problemas importantes en los métodos inductivos en lo que respecta a la indexación. Se deben adquirir muchos casos antes de que se puedan detectar índices relevantes (la inducción es demasiado lenta). También permite que se utilicen índices irrelevantes, y que se sigan utilizando hasta que se eliminen incrementalmente como resultado del propio proceso de inducción. Además, ocurrencias coincidentales pueden provocar la generación de índices erróneos. Esto nos conduce al problema final de cualquier método inductivo: la inducción sólo utiliza las evidencias disponibles al sistema. Sería mejor si la selección de índices se basara en teorías del dominio derivadas de principios lo más generales posible del funcionamiento del mundo, es decir, propiedades físicas, relaciones causa-efecto,etc... Este es el motivo del desarrollo de los sistemas EBL.

Uno de los autores que combinan la aproximación EBL con los sistemas tradicionales de CBR, es Hammond [HAM88]. Combina un sistema asociativo que relaciona un conjunto de síntomas predeterminado con las explicaciones pre-existentes, con un sistema que razona a partir de principios básicos mediante un razonamiento causal que permite construir explicaciones. Los sistemas que se basan en principios básicos deben rederivar cada nuevo diagnóstico, aunque se haya visto anteriormente un caso similar, y son sistemas muy lentos. En cambio, los sistemas asociativos son inflexibles ya que carecen del conocimiento necesario para tratar problemas que caen fuera de su contexto predefinido. Cuando se enfrenta con el problema de identificar el vocabulario de indexación surge la cuestión de como distinguir entre las características que realmente son importantes de las meramente coincidentales. Un programa con poco conocimiento del dominio no tendría base para poder decidir. Uno con un perfecto conocimiento del dominio sí, buscando a través de su base de reglas para determinar el valor predictivo de una característica. Sin embargo, considera que ambas aproximaciones son antinaturales en cualquier dominio realista de diagnosis. Propone utilizar el razonamiento causal no sólo para construir explicaciones, sino también para ayudar a relacionar características fácilmente identificables con los fallos que predicen. Dichas explicaciones pueden recuperarse rápidamente para su uso en casos futuros. Utilizando reglas causales y heurísticas acerca de las relaciones causales, el sistema puede hacer suposiciones acerca de la potencia predictiva de determinadas características que le permiten funcionar mientras espera que la posterior experiencia valide o niegue la validez de las relaciones

causales que ha supuesto.

Birnbaum [BIR88] afirma que la habilidad de un sistema para recuperar casos entre dominios depende, en parte, de su habilidad para derivar índices que reflejen las características importantes de un caso desde la perspectiva de un sistema de resolución de problemas. En algunos dominios las características importantes de las situaciones de problemas se centran alrededor de conceptos estratégicos independientes del dominio que son aplicables en múltiples dominios. Que el sistema aprenda conceptos estratégicos puede mejorar el rendimiento en diferentes dominios. Propone una forma de adquirir tales conceptos mediante el análisis de los fallos en la expectativa que produce un planificador. Utiliza técnicas EBL para la adquisición de nuevos conceptos estratégicos (totalmente nuevos o refinamientos de otros ya existentes). Otra conclusión a la que llega es que el conocimiento estratégico se extrae mejor de las explicaciones de los fallos en la expectativa. Este punto proporciona una motivación o una respuesta a la cuestión de ¿cuándo aprender un nuevo concepto?. Finalmente, identifica cuatro extensiones a los algoritmos típicos de EBL:

1. Construyendo explicaciones hipotéticas para creencias que no se justifican deductivamente, sino inductivamente, utilizando el contexto del problema para dirigir el razonamiento causal.
2. Considerando las metas del proceso de explicaciones para decidir qué vías de explicaciones posibles explorar.
3. Tratando con fallos en la expectativa universalmente cuantificados mediante la búsqueda de un ejemplo contrario específico.
4. Razonando hipotéticamente para determinar si la eliminación del factor que eliminó el fallo en la expectativa prevería de hecho dicho fallo.

Muchos autores se valen de técnicas EBL para generar conceptos abstractos independientes del dominio. Algunos ya se han comentado anteriormente como Koton [KOT89], o Kass [KAS89], y otros sobre los que se apoya este último son Ram [RAM89], y Navinchandra [NAV88].

Otras tendencias más recientes como [FOX94a] o [RAM94] utilizan respectivamente un aprendizaje introspectivo y un aprendizaje basado en metas. El primero utiliza como base un modelo de lo que sería un correcto funcionamiento del sistema (como cualquier model-based). El segundo combina un aprendizaje acerca del dominio mediante generalizaciones, el testeo de hipótesis o la reorganización de la memoria, con un aprendizaje acerca de su propio funcionamiento mediante el razonamiento introspectivo.

4 Métricas de Similaridad

La similaridad juega un papel importante en las teorías que intentan explicar como los humanos resuelven problemas, y consecuentemente en como deberían los sistemas de Inteligencia Artificial hacer lo mismo [BARE89a]. En general, los sistemas de IA estiman la similaridad mediante la comparación simbólica que describen situaciones (o clases generalizadas). Estas representaciones mediante características abstraen y evalúan los aspectos situacionales que son potencialmente relevantes para la meta actual del sistema de resolución de problemas. El grado importancia de las similaridades de las características predicen lo apropiado de la correspondiente acción recomendada. La valoración de similaridad definitiva es el grado en el que la acción recomendada que tuvo éxito en una situación pasada tiene éxito en la nueva.

En CBR, se resuelve un problema reconociendo su similaridad respecto a otro problema conocido (un caso) y transformando su solución para resolver la situación actual. La valoración de la similaridad afecta a todos los aspectos del razonamiento basado en casos:

- Similaridades entre las características resaltables del nuevo caso y características predictivas de casos pasados sugieren casos importantes a recuperar.
- Similaridades entre las características restantes del caso y las características predichas por el caso potencialmente relevante, pueden confirmar su relevancia.
- Diferencias que implican a características relevantes a la solución de un caso conocido guían la adaptación de la solución a la nueva situación.
- Diferencias que llevaron a fallos en la resolución disparan procesos de aprendizaje que se traducen en la asimilación de nuevos casos, en el refinamiento de la indexación, y en la adquisición de conocimiento adicional del dominio.

Una medida de la similaridad debe tener las siguientes propiedades [ALTH95]:

- Debe ser reflexiva: un caso es siempre similar a sí mismo.
- Debe ser simétrica: si un caso A es similar a un caso B, entonces el caso B es también similar al caso A[4]. En este sentido, la *similaridad* está relacionada con el concepto de *distancia*.

La distancia global entre dos casos se computa a partir de las distancias locales en la dimensión de atributos (características). La similaridad no siempre es transitiva. Es decir, si A es similar a B, y B es similar a C, no siempre podemos concluir que A es similar a C.

[4] [RIC95] es, quizás, una excepción a está regla, ya que propone una función de similaridad global no símetrica, que parte de una asimetría en la evaluación de la similaridad local entre dos características.

4.1 Similaridad Local y Global

La evaluación global de la similaridad entre dos casos se basa en la computación de la similaridad local entre cada atributo. La similaridad local puede variar, dependiendo del tipo del atributo o el tamaño de los conjuntos sobre los que se mide la similaridad. Por ejemplo, 10 es más similar a 20 si el tamaño del posible intervalo varía entre 0 y 1000, que si varía entre 0 y 20.

Las similaridades locales se combinan para generar una medida de similaridad global entre casos. Entonces, una similaridad global FS entre dos casos A y B descritos por *p* atributos se puede expresar de la siguiente manera:

$$FS(A, B) = F(Fs_1(a_1, b_1), Fs_2(a_2, b_2), ..., Fs_p(a_p, b_p))$$

Los métodos típicos de clasificación basados en el 'más cercano' (nearest neighbor) suelen utilizarse como función de similaridad [RIC95; MIY95; WAT94], algunos de los cuales se muestran en la tabla 1. Estos métodos presentan la ventaja de soportar el aprendizaje de nuevos casos sin que se degrade el rendimiento sobre los datos de entrenamiento previos. Más aún, trabajan con un orden de magnitud menor de parámetros necesarios que los métodos back propagation o radiales, y son bastante sencillos de implementar. Sin embargo, también poseen algunos aspectos negativos que limitan su aplicabilidad. En general, poseen rendimientos pobres en la generalización, son sensibles a existencia de características ruidosas, necesitan grandes conjuntos de entrenamiento, y el tiempo de ejecución aumenta con el tamaño del conjunto de entrenamiento. Muchos de estos inconvenientes han sido abordados en algunos trabajos, los cuales serán parcialmente comentados en esta sección.

Similaridad Global	Nombre
$FS(A,B) = \frac{\sum_{i=1}^{p} \omega_i \, Fs_i(a_i,b_i)}{\sum_{i=1}^{p} \omega_i}$	Nearest Neighbor
$FS(A,B) = \frac{1}{p} \sum_{i=1}^{p} Fs_i(a_i,b_i)$	Block City
$FS(A,B) = \sum_{i=1}^{p} \omega_i \, Fs_i(a_i,b_i)$	Weighted Block City
$FS(A,B) = \frac{1}{p} \sum_{i=1}^{p} \sqrt{(Fs_i(a_i,b_i))^2}$	Euclidean
$FS(A,B) = \frac{1}{p} \sum_{i=1}^{p} \sqrt[r]{(Fs_i(a_i,b_i))^r}$	Minkowski
$FS(A,B) = \sum_{i=1}^{p} \omega_i \sqrt[r]{(Fs_i(a_i,b_i))^r}$	Weighted Minkowski
$FS(A,B) = \max_i \omega_i \, Fs_i(a_i,b_i)$	Maximum

Tabla 1

4.2 Tendencias

Las tendencias de investigación en lo que a la valoración de similaridad se refiere pueden resumirse en los siguientes puntos:

1. Se diferencia entre recuperación de casos y valoración de la similaridad. Aunque pueden recuperarse casos apoyándose en las expectativas de similaridad, el proceso de recuperación es poco restringido y abarca demasiados casos. Posteriormente se necesitaría una estimación de la similaridad para determinar el caso que más se asemeja a la nueva situación.
2. Los casos almacenados y las situaciones de resolución de problemas se representan en términos de características abstractas. La extracción de características a partir de datos de bajo nivel es externa al sistema.
3. Las representaciones de casos equivalentes no son siempre uniformes cuando se describen en términos de características equivalentes. Tal falta de uniformidad necesita de la inferencia de una equivalencia de características en lugar de una similaridad (matching) directo.
4. Es necesario conocimiento del dominio, además de las características del caso, para poder razonar acerca de similaridades. Tal conocimiento permite la inferencia de la equivalencia de características y la evaluación de la importancia de características no reconocidas durante la valoración de la similaridad.
5. Se presupone un único contexto de resolución de problemas.
6. Se emplea heurística o funciones de similaridad numérica para imponer una ordenación parcial sobre los casos recuperados, basándose en las similaridades de sus características en relación con el nuevo caso.

4.3 Cuestiones fundamentales en la valoración de la similaridad

Estas son resultado de una investigación realizada por King en 1989 (citado en [BAREa89]) entre investigadores de IA, psicólogos, expertos en recuperación de información, y otros. De esta consulta se formularon estas siete cuestiones:

¿Por qué se percibe una situación como similar a otra? Esta es la esencia de la investigación. Parece que 'la razón' por la que una situación es similar a otra es altamente dependiente del dominio de la aplicación. De aquí surge la cuestión de si existen una serie de principios generales aplicables a la valoración de la similaridad.

¿Que relación existe entre la valoración de la similaridad y la recuperación de casos? La recuperación de casos es el primer paso del proceso de valoración. El proceso de recuperación devuelve una serie de casos. Entre estos hay que elegir el más similar. La valoración de la similaridad también guía el aprendizaje de la estructura de indexación sobre la que se apoya la recuperación de casos. Otra cuestión derivable sería cómo se distribuye la valoración de la similaridad entre subprocesos.

¿Puede representarse un caso sólo con características de bajo nivel? Puede que las características necesarias para la resolución del problema impliquen relaciones complejas. ¿Este esfuerzo de abstracción se realiza durante la representación de los casos o durante la resolución del problema?.

¿Cómo se valora la similaridad cuando los casos no se representan uniformemente? Este problema aparece cuando el sistema es dependiente del contexto. Consecuentemente, se debe inferir la equivalencia de características durante la resolución de problemas en lugar de durante la compilación de la representación de los casos.

¿Qué papel juega el conocimiento general del dominio en la valoración de la similaridad? Todos los investigadores coinciden en la necesidad de la presencia de este conocimiento en los sistemas CBR. El tipo, localización, cantidad, y adquisición de este conocimiento aún está por determinar.

¿Cómo influyen los contextos de la resolución de problemas sobre la valoración de la similaridad? Normalmente se utiliza una única base de conocimiento, aunque las nuevas tendencias de AI sugieren la utilización de distintas bases. Hasta la fecha del estudio no se había hecho ningún esfuerzo en ese sentido en los sistemas CBR. Un tema abierto es como el contexto determina la identificación y evaluación de características.

¿Se puede computar la similaridad o simplemente se conoce de las experiencias pasadas? Se supone que se puede computar la similaridad para elegir la experiencia más similar.

4.4 Tipos de Similaridades

Whitaker [WHI89] propone una categorización de los distintos tipos de similaridades que pueden encontrarse en un sistema:

1. *Sinónimos.* La utilización de sinónimos en las descripciones de los casos puede impedir la selección del mismo como similar a la situación nueva. La herramienta

CBR Express sustituye los sinónimos que encuentra en la descripción de consulta por palabras básicas que se le han suministrado previamente al sistema. Lo que no contempla es el contexto en el que un sinónimo *no* debe ser sustituido.

2. *Categorías Ordinales.* Supongamos que se pudieran dividir los casos en categorías ordinales. Se podrían considerar los casos de categorías adyacentes como más similares que los lejanos.
3. *Análisis del Perfil.* Si se almacenara un caso con un formato de marco, se podrían utilizar los distintos atributos del caso para recuperarlos individualmente por alguno de los métodos anteriores, o se podría reconocer un determinado 'perfil' entre el caso nuevo y los casos de la base. En el caso de campos numéricos se podría hacer pesando cada atributo, y tomando la diferencia entre cada característica del caso de consulta y de los casos de memoria. La suma de las diferencias será el criterio de selección. El menor número indica la mayor similaridad. ¿Cómo se procedería en el caso de atributos no numéricos?
4. *Algoritmo de agrupamiento.* Existen múltiples formas de agrupar conjuntos de datos cuando se conocen los valores de sus atributos. De igual forma se podría considerar un caso
5. *Cualificadores.* Cuando existen situaciones cuya interpretación es dependiente del contexto, se hace necesario el uso de cualificadores que nos permitan discernir entre las distintas opciones.
6. *Utilización de reglas generadas del análisis de la base de casos.* Estas reglas poseen una capacidad predictiva acerca del comportamiento de casos similares en la base de casos.
7. *Código Modificable.* Consiste en permitir que las reglas de similaridad cambien con la experiencia del usuario o por la adición de casos a la base de conocimiento.

4.5 Pasos en la Valoración de la Similaridad

En línea con lo comentado anteriormente acerca de que la recuperación es sólo el primer paso de la valoración de la similaridad, Ashley [ASH89] propone una serie de pasos a seguir en la valoración de la similaridad entre dos casos, y comenta una serie de problemas relacionados con este proceso:

1. Definición de aquellas características que son importantes reconocer en el caso de búsqueda y en los casos de memoria, y del modelo que justifique porqué son importantes dichas características.
2. Método, dado un problema, para determinar aquellos casos que reconocen (match) características importantes del problema.

3. Método, dado un caso reconocido, de valorar la significancia de las características importantes que se reconocen y las que no entre el caso y el problema.
4. Método, dado un caso reconocido, de valorar si existen otros casos más similares al problema que el caso.

El primer requisito, definir las características importantes reconocer, está relacionado con qué representar en un caso. Las características que son importantes son las que están relacionadas con las salidas de los casos. Cuando se elijan, basta con saber las consecuencias posibles de los casos y si una determinada característica favorece o no favorece una consecuencia en particular. No es necesario que exista una detallada cadena causal de inferencia que relacione características con consecuencias.

Como se indica en el tercer y cuarto requisito, la valoración de la similaridad entre un caso particular *a* y el problema *p* requiere considerar y existen otros casos que son más similares a *p* que *a*. Si existiera algún otro caso *b* más similar a *p* que *a*, sería crucial considerar este caso *b* en lugar de, o, al menos en suma a, *a*, especialmente si las consecuencias de *b* son distintas de las de *a*. Esta situación hipotética se podría representar utilizando diagramas de Venn tal como muestra la siguiente figura 4. Un caso o un problema se representa como un subconjunto del universo de todas las características importantes al sistema.

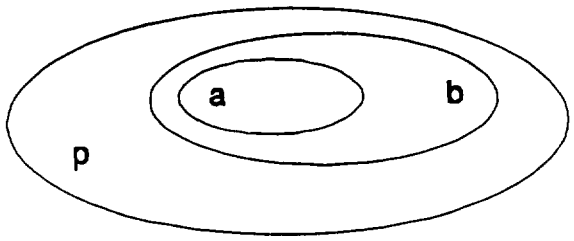


Figura 4

Las dificultades surgen cuando *a* o *b* tienen características importantes que no comparten con el problema o con el otro caso, o el subconjunto de relaciones no es aplicable al conjunto de características importantes que comparten los casos *a* y *b* con el problema. La figura 5 muestra algunos ejemplos de comparaciones difíciles. En definitiva, los casos no son comparables, al menos sin un análisis más profundo.

Sin embargo, es importante comparar casos 'incomparables', especialmente si tienen consecuencias distintas. En estas situaciones hay que centrarse en las diferencias importantes que tienen con respecto al problema y entre si. Si se pudiera saber cual de estas diferencias está más relacionada con las consecuencias del caso, se tendría una base para poder elegir entre los casos incomparables.

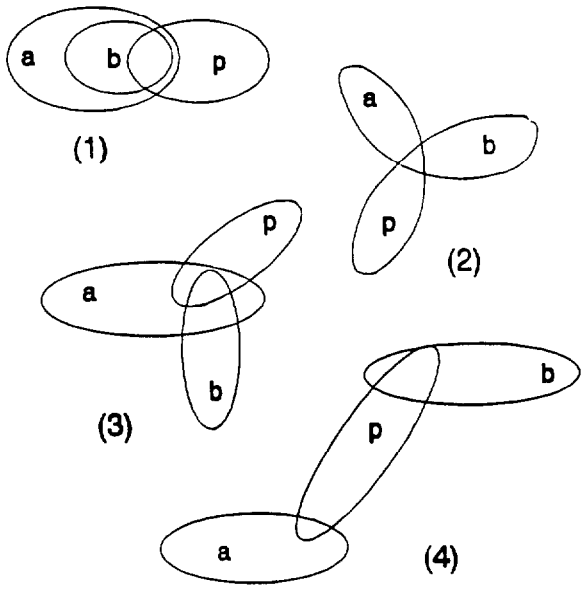


Figura 5

4.6 Métodos Básicos

Los cuatro métodos básicos para valorar la importancia relativa de características en el análisis de casos son: manuales, estadísticos, analíticos, y basados en precedentes.

Algunos sistemas como Battle Planner [GOO89] y CASPIAN [CAS95] exigen que sea el ingeniero del conocimiento, supuestamente guiado por el experto, el que indique la relevancia relativa de cada atributo definido en el sistema

En algunos dominios se podrían utilizar el análisis estadístico de múltiples casos para asignar un peso numérico a las características y así discriminar entre ellas. Alternativamente, en otros dominios existe un modelo causal analítico de la tarea de resolución del problema que puede ser suficiente para inferir la importancia relativa de las características. Por ejemplo, podría existir una jerarquía de metas en la que se relacionan las características, de forma que se tenga preferencia por aquellas características conectadas con las metas más abstractas.

Donde no sean efectivos los modelos analíticos o estadísticos, aún queda otra oportunidad de asignar pesos a las características mediante la referencia a casos precedentes que aportan información acerca de la significancia de las características. La idea general consiste en que dado un problema con algún conjunto de características conflictivas, encontrar un caso que tuviera el mismo conjunto de características conflictivas y argumentar que se pueden deducir

las mismas consecuencias. Para discriminar entre características que no son conflictivas, encontrar dos casos con consecuencias opuestas que son las mismas excepto para el hecho de que cada una tiene una característica diferente. En los argumentos se utilizan distinciones, la cita de contra ejemplos, y la proposición de hipótesis para demostrar las limitaciones de un precedente como evidencia de la importancia relativa entre factores competitivos.

Uno de los trabajos que utiliza una combinación de esta última aproximación con un modelo analítico para valorar similaridades es el de Golding [GOL89]. Este combina un sistema basado en analogías y similaridades. El método analítico invoca una teoría del dominio para justificar el por qué una solución pasada es aplicable en una nueva situación. Esta aproximación puede evitar la generación de analogías sin sentido siempre y cuando se disponga de una teoría de justificaciones completa y correcta. Los sistemas basados en similaridades eligen la fuente de analogías (el caso preseleccionado) que mejor reconozca el objetivo utilizando alguna métrica de similaridad.

El criterio analítico consiste en que la regla de control de la búsqueda que sugirió el operador en el problema fuente también es aplicable al objetivo. Cuando se verifique el criterio, el problema objetivo tendrá la misma respuesta que el problema fuente y por las mismas razones. El criterio basado en las similaridades tasa la similaridad entre la fuente y el objetivo que determinan la solución fuente. Aunque de desconozca el conjunto preciso de características relevantes, el núcleo analítico proporciona una aproximación, y también un límite superior (por la premisa de circunscripción). Las características en el núcleo analítico determinan la solución fuente, en el sentido de hasta qué punto es necesaria la teoría del dominio. El criterio basado en la similaridad proporciona la mayor importancia de dichas características. La similaridad entre una fuente y un objetivo es función no sólo de las características que se ajustan entre los dos problemas, sino también de las características del núcleo analítico. Cada núcleo tiene una métrica de similaridad distinta, por lo que las dependencias del núcleo se evalúan separadamente.

Dos sistemas que prefieren la tercera aproximación (apoyarse en justificaciones precedentes) son GREBE [BARE89a; BRAN91] y PROTOS [PRO89]. GREBE utiliza las explicaciones para valorar la similaridad de un nuevo caso con respecto a uno precedente. Se supone que ambos son similares en la medida en que sean aplicables las explicaciones de relaciones clave en el caso de memoria sobre los hechos del nuevo caso. Las explicaciones son los propios casos. La valoración del grado de similaridad entre el nuevo caso y un ejemplo con respecto a una conclusión dada se realiza primero intentando mapear el subgrafo que representa a los hechos criterios del ejemplo sobre el nuevo caso. Se realiza una búsqueda mejor-el-primero para los posibles mappings entre los hechos criterios el ejemplo y el nuevo caso, utilizando las menores triplas no reconocidas como función de evaluación. Entonces GREBE se llama recursivamente para intentar inferir cualquier hecho perdido del nuevo caso

que se necesita para un reconocimiento perfecto. Los hechos perdidos pueden inferirse rehusando la explicación de ejemplos previos. Dichas explicaciones pueden ser o Generalization Based Exemplars o Explanation Based Exemplars.

PROTOS valora la similaridad de una nueva situación respecto a un caso pasado mediante explicaciones de como sus características proporcionan una evidencia equivalente para una clasificación. Estas explicaciones de equivalencia de características se formulan mediante una búsqueda heurística a través de una red de conocimiento del dominio obtenida de un Experto. La importancia de las características que no se reconozcan se determina mediante la evaluación de las explicaciones de importancia en la clasificación del caso. La valoración global de similaridad se basa en la evaluación heurística de la calidad de la explicación y de la importancia de las características no reconocidas. La representación de la similaridad en PROTOS es bastante simple por dos motivos: primero, existe un único contexto para determinar la similaridad; lo casos son similares respecto a una categorización de la diagnosis. Segundo, la similaridad global se evalúa mediante la 'suma' de las similaridades de las características. Un caso se representa por un vector de características, y una explicación consiste en un camino de inferencia heurístico que relaciona dos características, normalmente por medio de abstracciones comunes.

El proyecto GREBE, al igual que PROTOS, utiliza un único contexto para determinar la similaridad. Sin embargo, un caso es una red compleja que codifica una historia (precedentes jurídicos), y tratar de compilarla en un vector de características sería impensable. Aún más, una explicación de similaridad necesita justificaciones complejas, y no lineales y una cantidad importante de conocimiento del dominio. Debido a la complejidad de las explicaciones, un objetivo de la investigación en este proyecto consiste en la reutilización de explicaciones de similaridad. Esto es importante, ya que casos pasados de valoración de similaridad pueden sustituir conocimiento del dominio.

Porter en este mismo artículo enumera los tipos de conocimientos que debería presentar una representación de la similaridad entre dos casos dependiente del contexto:

- El o los contextos en los que son similares.
- La base para la similaridad.
- Los límites de la similaridad (en qué sentido difieren los casos).
- Usos (y malos usos) de la similaridad.
- Similaridades relacionadas, como generalizaciones y especializaciones.

4.7 Otros trabajos representativos

Una vez comentados una serie de aspectos importantes y comunes en la valoración de la similaridad, pasaremos ahora a comentar las aproximaciones utilizadas por algunos de los sistemas más importantes. La base de la discusión se extrajo del artículo de Bareiss y King [BARE89a], aunque ha sido completada con artículos de los propios autores de los sistemas, y con otros trabajos de interés.

PROXIMITY, GROTH, y SHRINK de Kibler y Aha proporcionan un estructura para evaluar el rendimiento de estrategias complejas de valoración de similaridad. Cuando se presenta una nueva situación, comparan las descripciones de sus características con las de cada caso conocido, y les asigna una puntuación de similaridad calculada contando características reconocidas. Estos sistemas están limitados por la suposición de que es posible realizar una búsqueda exhaustiva y que existe una descripción uniforme para las características.

En CYRUS, SHRINK, y JULIA de Kolodner la valoración de la similaridad se combina con el proceso de indexación. Los casos recuperados durante el recorrido de la jerarquía de indexación ya se sabe que son similares porque los casos se reconocen sobre las características de indexación. Sin embargo, los reconocimientos de las características pueden no ser directos porque las características del nuevo caso pueden haberse transformado (utilizando conocimiento de fundamentos estático) para determinar los índices. La ocurrencia de las transformaciones de las características no afecta contrariamente a la valoración de la similaridad; el reconocimiento que implica transformaciones tiene el mismo status que el reconocimiento directo. Cuando el caso recuperado se demuestra inadecuado para la resolución del problema, los sistemas SHRINK y JULIA utilizan una serie de índices secundarios basados en fallos. Cuando se recupera un caso por medio de tal enlace, se supone que tiene una mayor similaridad al nuevo caso porque se vio implicado en un fallo similar.

El sistema PARADYME de Kolodner [KOL89], fue diseñado para ayudar a su otro sistema JULIA. La filosofía de este sistema consiste en encontrar aquellos casos que mejor cooperen en la resolución de la meta *actual*. Así, primero construye una lista de casos similares en función de un reconocimiento parcial, para después aplicar una heurística de preferencia al conjunto recuperado para saber cual es el mejor. La búsqueda se realiza utilizando características 'preferentes', y si falla busca con el resto de características. Las características preferentes son las que se demostraron útiles en el pasado.

PRODIGY [HAI95a; HAI95b] utiliza una función dependiente de la tarea a realizar para evaluar la similaridad entre dos situaciones. La tarea es la de la planificación de rutas, y la función se basa en técnicas de triangulación de grafos.

PARADYME utiliza seis heurísticas de preferencia para seleccionar el mejor caso. Ordenadas según su prioridad (orden en el que se aplican) son las siguientes: dirigida por metas (goal-directed), característica preferente (salient feature), especificidad, frecuencia, reciencia (de reciente), y facilidad de adaptación.

La primera consiste en preferir aquellos casos que ayudan a resolver las metas actuales, y de ellos, los que compartan más restricciones. Con la segunda heurística se prefieren aquellos casos que reconocen características preferentes sobre los que reconocen otras características, y preferir los que reconocen más características sobre los que reconocen menos. La heurística de especificidad se basa en que un reconocimiento más específico (ej. más abajo en una jerarquía de abstracción) predice más que uno más general. Las características del caso menos específico son un subconjunto del caso más específico. Las dos heurísticas siguientes prefieren los casos que se han utilizado más frecuentemente y más recientemente respectivamente. La última establece que en caso de duda elegir el caso que sea más fácil de adaptar a las circunstancias actuales.

Pero, ¿qué son las características preferentes? Se podrían agrupar en tres tipos de índices:

- Goal Achievement: predicen la aplicabilidad de algún método para alcanzar una meta. Son conjunciones de metas, restricciones sobre dichas metas, y características que predicen el método o solución para alcanzar la meta o conjunto de metas.
- Solution Evaluation: predicen el éxito o fracaso de una solución. Son conjunciones de características que predicen consecuencias para usuales (fallos, éxitos inesperados, efectos colaterales...). Son útiles para evaluar una solución que ha sido propuesta.
- Outcome Achievement Sets: describen consecuencias inusuales. Son conjunciones de características que describen consecuencias poco usuales. Son útiles cuando se trata de explicar una situación anómala, y cuando se conoce la forma de una solución pero no se sabe como alcanzarla.

Cualquier caso puede tener varios conjuntos de características conjuntivas asociadas al mismo. Podría tener un conjunto para cada meta alcanzada en el curso del razonamiento del caso.

El sistema MEDIATOR de Simpson busca todos los caminos de indexación posibles y normalmente recupera múltiples casos. Después de la recuperación de un caso, un procedimiento heurístico tasa los casos recuperados de acuerdo su valoración de similaridad respecto al nuevo caso. Este procedimiento elimina primero todos los casos en los que no sean idénticas las características más importantes con respecto al nuevo caso. Los casos restantes se tasan según sea el número de reconocimientos sobre un subconjunto

(preseleccionado) de características que se suponen las más importantes. Si en este punto no se pudiera tomar una decisión, se seleccionará un caso según un esquema de preferencias prefijado. Al igual que MEDIATOR de Kolodner utiliza reconocimiento transformacional de características de casos a índices. Durante el proceso de valoración del reconocimiento de las características, el número y tipo de tales transformaciones afectan a las tasas de similaridad. Al igual que SHRINK y JULIA, hace uso de un procedimiento de indexación secundario basado en fallos durante la resolución de problemas.

A diferencia de los otros sistemas, el sistema HIPO [ASH86; RIS88] de Ashley y Rissland no se limita a seleccionar un único mejor caso, sino que selecciona todos los casos similares. HIPO recupera todos los casos que se reconocen completamente (match) o parcialmente (near match) el nuevo caso en alguna dimensión. Los casos que tienen dimensiones en apoyo de la posición que se está argumentando y ninguna en contra se consideran como los 'most on point'. La significancia de una dimensión depende del contexto del argumento. El contexto se caracteriza por los factores específicos de un caso y el papel que juega un caso en un argumento.

Cuando se presenta una nueva situación, HIPO construye un 'registro de análisis del caso' que contiene los predicados factuales aplicables, las dimensiones aplicables, las dimensiones near-miss, demandas potenciales, y casos relevantes de la base de casos. Con esta información HIPO construye una 'claim lattice', que consiste en un árbol en el que se sitúa como nodo raíz la situación actual del caso (current fact situation) junto con una lista de las dimensiones aplicables y las reconocidas parcialmente. Estas últimas son dimensiones en las que se satisfacen la mayoría, pero no todos, los requisitos. Los hijos de este nodo raíz serán casos que comparten un subconjunto de las dimensiones del nodo raíz. Los casos que más dimensiones comparten con el nodo raíz estarán más cerca de dicho nodo que aquellos que tengan pocas dimensiones en común. Los casos mop (most on point) son aquellos hijos inmediatos al nodo raíz que no tienen near-miss dimensiones. Con toda esta información

HIPO compara y contrasta casos a tres niveles: hechos, justificaciones, y argumentos. A nivel de hechos, se compara la situación actual del caso con los casos importantes de la claim lattice centrándose en los hechos importantes que comparten tal como indican las dimensiones que tienen en común. De esta forma se construyen analogías entre la situación actual del caso y distintos casos de memoria. En el nivel de justificaciones, se comparan entre si casos importantes utilizando la claim lattice para ver cual es el mejor precedente a la situación en estudio. Los casos se comparan en términos de: cuánto de importante son en relación a la situación actual del caso (en que medida son mops), que utilidad tienen como argumento legal en la situación actual, y que utilidad potencial tendrían como argumento legal en la situación actual encontrando casos en favor de la postura contraria que puedan utilizarse para distinguir mops. En el nivel de argumentos, se utilizan las claim lattices para evaluar los argumentos

en favor de una proposición, esencialmente mediante la comparación de las relaciones de los mop en favor y en contra.

El sistema CASEY de Koton utiliza un modelo causal predefinido para inferir la importancia y la equivalencia de las características. CASEY abstrae el estado patológico subyacente (es un sistema de diagnosis médica) a partir de los síntomas actuales para realizar el reconocimiento. Realmente se utilizan los estados de memoria (una combinación de síntomas) en lugar de utilizar los síntomas directamente.

Otra aproximación importante es el sistema de control continuo de Lopeikina [LOP88]. Una vez recuperado un conjunto de casos candidatos, el 'selector' los poda. Los casos candidatos deben ajustarse perfectamente a las características de indexación cruciales para que puedan ser recuperados. Las distinciones más 'finas', que permitirán reconocimientos parciales, las hará el selector. La tarea del selector es la de realizar un ajuste más fino entre la situación actual del problema y los casos candidatos y determinar el más relevante.

¿En qué medida debe un sistema seleccionar un caso? un selector simple elegiría rápidamente e imprecisamente, y dejaría que el procesamiento posterior rechazase las elecciones pobres. Un selector más sofisticado dedicaría más recursos a intentar seleccionar los mejores casos (aunque la decisión final del caso esperaría el éxito del procesamiento siguiente). Esta última aproximación, que es la que utiliza Lopeikina, es más útil porque los rechazos pueden dar lugar a acciones erróneas y a pérdidas de tiempo.

El proceso de selección implica dos pasos. Primero la evaluación de la calidad del matching. Y posteriormente una estimación del éxito o fracaso. Durante la evaluación de la calidad del ajuste se buscan similaridades y diferencias y se valoran sus importancias. Existen dos niveles de reconocimiento (matching):

- A nivel de variable lógica: significa que el caso que se reconoce habla de los mismos conceptos que la definición del problema. Se utiliza cuando el problema se compara con caso genérico.
- A nivel de valor: en este caso hay que filtrar los datos ruidosos utilizando un 'rango de sensibilidad', el cual define un matching para una característica.

La importancia de un reconocimiento o de una diferencia se traduce por un peso, y es específico para cada caso; una característica puede ser crucial para unos casos e irrelevante para otros. Los pesos son determinados a priori.

Las características se organizan en tres categorías de importancia para cada caso:

- No Importante: el no ser capaz de 'reconocer' las características de este grupo no inhabilita para seleccionar el caso.
- Importante: se compone de entradas que deben reconocerse para que el caso sea útil para resolver el problema actual. Si se detecta alguna diferencia el 'Modificador' posee el conocimiento necesario para modificar el caso para que se ajuste a la situación actual.
- Muy Importante: definen la aplicabilidad del caso. Si existiera alguna diferencia en esta categoría el caso se rechazaría. Estas características se utilizan para construir índices.

El proceso para asignar una característica en una categoría de importancia se apoya en ciertas consideraciones como la cantidad de trabajo que se necesitaría para modificar el caso para ajustarlo a la nueva situación.

El Selector también considera lo apropiado de las consecuencias que se experimentaron con el caso almacenado. Esta aproximación favorece los casos que son probables que conduzcan a los mejores resultados, e identifique qué casos importantes con resultados desfavorables deben ser especialmente vigilados. La información contenida en el caso controla la función de evaluación de éxito. Esta información de consecuencias deseables será utilizada por el Monitor. Este se encarga del seguimiento de las acciones emprendidas.

El trabajo de Weber [WEB95] utiliza una función Fuzzy para evaluar la similaridad y recuperar el mejor caso de memoria. La función fuzzy asigna etiquetas difusas a determinados parámetros del sistema. El algoritmo de recuperación buscará entonces casos con etiquetas iguales. Estos casos pre-recuperados serán posteriormente filtrados mediante la aplicación de otra función difusa. El inconveniente, característico de este tipo de lógica, es que los parámetros de la función fuzzy deben definirse empíricamente, y todavía están en ello.

El último sistema a comentar es OGRE de Donahue [DON89]. OGRE representa un caso como un conjunto de factores, y cada factor con un conjunto de características como nombre, tipo, valor por defecto, el método de comparación entre dos valores... Para determinar la similaridad entre dos experiencias, examina cada factor común a ambas experiencias, y las compara de acuerdo al método especificado en el descriptor correspondiente del factor. OGRE dispone de un conjunto de funciones genéricas de comparación, o el experto puede diseñarse su propia función de comparación.

Después de determinar el valor de comparación para dos valores de un factor, se ajusta dicho valor mediante la aplicación del peso del descriptor. El peso de un factor puede ser un

número, lo que implica una manera directa de ajuste.

Ya que la importancia de un factor puede depender del contexto del caso, el peso puede depender funcionalmente del valor de la comparación. El valor definitivo devuelto después de aplicar el peso del factor al resultado de la comparación se denomina *similaridad pesada del factor*. La similaridad total entre dos experiencias es la suma de la similaridad pesada del factor de cada factor de los casos.

En la mayoría de los casos, los valores de los factores del caso se proporcionan por el usuario, con las restricciones proporcionadas por el descriptor del factor. Sin embargo, en algunos casos un factor puede necesitar obtener su valor de otros factores de la situación (por ejemplo, la densidad puede depender de los valores de masa y volumen). El experto en el dominio puede expresar las interdependencias del valor del factor sobre otros factores mediante la asignación al descriptor del factor de una función de valores computados.

Durante la recuperación el sistema accede a los casos almacenados en la librería, y calcula una similaridad para cada experiencia respecto a la situación nueva. Los casos se ordenan por similaridad, y los resultados se presentan al usuario para que este los interprete.

5 Adaptación de casos

Una vez hemos seleccionado el caso que mejor refleja la situación actual se hace necesaria la adaptación de las características del mismo a las circunstancias actuales. Este proceso normalmente vendrá acompañado de dos efectos: primero se lanzará un procedimiento de validación de la adaptación; y segundo puede que se tengan que hacer cambios sobre la memoria para ajustarla a las situaciones actuales, ya que como señala Callan en [CAL91] 'normalmente se guardan demasiados casos, lo que nos conduce a resaltar la importancia de los mecanismos de adaptación y recuperación', o a realizar un análisis de la utilidad de los episodios almacenados [FRA94].

A pesar de que las tareas de los sistemas CBR son muy variadas, el estudio de la adaptación de casos se ha centrado sobre tres tareas: planificación, explicaciones, y diseño. Cada una de estas tareas tiene sus propios requisitos funcionales y métricas de evaluación, y como resultado, cada una define su propio espacio de posibles técnicas de modificación. Por ejemplo, la planificación tiene un alto componente de ejecución sobre la línea de construcción y ejecución de un plan. Por tanto, la modificación de planes no puede separarse fácilmente de la ejecución de planes. Las tareas de diseño, por otro lado, rara vez producen resultados que se evalúan mediante la ejecución. Esto significa que se debe utilizar alguna otra métrica para poder decidir si una solución de diseño puede aceptarse o rechazarse. De igual manera, con las tareas de explicaciones, la modificación de una explicación debe conducirse mediante una evaluación de la bondad con la que la explicación actual encaja en el conjunto de restricciones existente.

Las aproximaciones a la adaptación también difieren en la dimensión del tipo de conocimiento utilizado por el proceso de modificación. Algunos sistemas utilizan un conocimiento muy profundo de las dependencias en un plan o las reglas utilizadas para formar una explicación inicial, mientras que otros hacen uso de técnicas con un nivel más superficial para modificar sus casos.

Una tercera diferencia reside a nivel de la dependencia del dominio mismo. Por ejemplo, en los trabajos de Hinrichs [HIN89; HIN91], las técnicas de modificación poseen un dominio de independencia para ellas, mientras que Kass [KAS89] y Goel [GOE89] utilizan técnicas que están más íntimamente ligadas tanto a la tarea como al dominio. El trabajo de Converse [CON89] refuerza la utilización de un conjunto inicial de reglas neutrales del dominio que se incrementarán en el tiempo con las reglas del dominio aprendidas a partir de las explicaciones de fallos en la expectativa. Collins [COL89] tiene una aproximación diferente en la que intenta construir transformaciones que optimizan planes sobre un conjunto de meta-

metas válidas en cualquier dominio. En definitiva, parece que un conocimiento detallado del dominio no es imprescindible para hacer los ajustes, aunque los sistemas que mejor se comportan son aquellos que contienen características híbridas como el trabajo de Fuchs [FUC95].

A pesar de todas estas diferencias, los distintos modelos de adaptación difieren poco conceptualmente. La mayoría se apoyan de una forma u otra en operaciones básicas como la sustitución, eliminación, concatenación, etc..., aunque sobre operadores distintos (subplanes, subdiseños, XP's, etc...).

No basta con saber en cada momento cual es el operador de modificación que hay que utilizar; muchas veces se olvida que primero es necesario encotrar sobre qué porción de información se va a aplicar la modificación. Es aquí, a la hora de encotrar diferencias significativas entre dos casos para poder adaptarlas, donde fallan la mayoría de los algoritmos que implementan la analogía derivacional como se expone en [BLU94].

Más recientemente, Leake [LEA94; LEA95] se vale de una aproximación de CBR para guiar al proceso de búsqueda de qué modificar, y que operador aplicar para la adaptación de casos. Mediante lo que denomina introspección busca en su base si existe un plan que le permita adaptar la situación actual, y en caso negativo construye uno nuevo utilizando reglas genéricas de adaptación para guiar la búsqueda. Su propuesta, sin embargo, presenta dos inconvenientes, el primero es el de la validación del plan que ha construido, y el segundo el de verificar la causa de un fallo en la búsqueda (si es debida a un vacío en el conocimiento, o a un plan defectuoso. Aspecto este importante, ya que su estrategia consiste en almacenar todo plan que construya, lo cual puede dar lugar a inconsistencias en el conocimiento que almacena.

Las siguientes páginas resumen brevemente algunas de las aproximaciones anteriores.

5.1 Collins

Considera especialmente costoso diseñar un plan desde cero. Si se pudiera adaptar un antiguo plan a una nueva situación, se eliminarían gran cantidad de los costes asociados con la generación de planes. Si este plan modificado también se almacena y además sirve para mejorar los planes de una 'clase' de situaciones, el planificador accederá al plan modificado en todas esas situaciones, mientras se tenga un correcto mecanismo de indexación. Collins propone la existencia de un conjunto de reglas de transformación independientes del dominio para los planes.

Si no existiera ningún problema para aplicar un plan, el planificador sólo tendría que aplicarlo; la adaptación consistiría en la identificación de un problema en la aplicación de un plan, y realizar los cambios que eviten tal problema. Existen dos formas de *copiar* lo que se ha hecho con un plan en otro. En una, se aislan la partes individuales del viejo plan que son responsables del problema, y estas se eliminan o se reemplazan, mientras que la estructura global del plan queda intacta. En la otra, se dejan intactos los módulos individuales del viejo plan, pero se reorganizan en una estructura global diferente. En estos casos, la adaptación de planes implica una reorganización global, más que el aislamiento y sustitución de pasos particulares. ¿Cómo sabe el planificador que con una determinada reorganización se conseguirá la adaptación deseada, sin tener que desechar la estructura original y replanificar desde cero?

Considera que el modelo más plausible para tal proceso implica un conjunto de reglas de transformación independientes del dominio, las cuales especifican que los planes que reconozcan un patrón abstracto pueden transformarse de una manera determinada para producir planes alternativos que reúnan las mismas metas pero las ejecuten de distinta forma respecto a algunos de los inconvenientes implicados en la elección el plan óptimo:

- Una posible transformación sería un tipo de factorización: elementos similares de planes alternativos se mezclan, permitiendo que se ejecuten por anticipado a la decisión entre las alternativas. Esto proporciona al planificador más tiempo para poder tomar una decisión, lo cual es normalmente una mejora. Esta ventaja tiene, sin embargo, un inconveniente, contra la pérdida de habilidad para optimizar los pasos combinados para la alternativa específica que se utilizará.
- Otra posible transformación posible implica la modificación de una cantidad hasta que se ajuste al valor objetivo. Un plan que basado en un continuo seguimiento de una cantidad se sustituye por uno que implica un seguimiento continuo, en la que el planificador realiza una aproximación inicial de como alterar la cantidad, ejecuta la suposición, observa lo lejos que quedó la suposición, y realiza una nueva suposición. Esto libera al planificador de la necesidad de realizar un seguimiento continuo de la situación, lo cual usualmente es una ventaja. Por otra parte, se generan dos inconvenientes posibles: Primero, una aproximación puede sobrepasar la marca, lo cual es una ventaja en situaciones donde es mucho más costoso ajustar la cantidad en un sentido que en el otro, por ejemplo, cuando se saca líquido de una botella; segundo, no existe en el tiempo máximo que podrían tomar la aproximaciones sucesivas en el peor de los casos, si las aproximaciones son pobres, cuando el seguimiento continuo produciría el resultado en un número fijo y predecible de unidades de tiempo.
- Otra aproximación sustituye una toma de decisiones por otra que optimiza el número de opciones abiertas. Esto es especialmente útil a la hora de hacer frente a sucesos inesperados.

- Otra posibilidad consiste en sustituir un plan para adquirir elementos de un recurso basados en la identificación de la categoría del recurso que se necesita, por uno que ignora la categoría y optimiza la calidad total del recurso, dependiendo de un ajuste entre la distribución de recursos disponibles y la distribución de las necesidades de asegurar que todos los recursos se utilizan. El compromiso es entre ganancias a largo plazo versus corto plazo. Siempre es mejor utilizar todos los recursos a corto plazo, a largo plazo normalmente es mejor adquirir el mejor elemento disponible.

Collins ha identificado un conjunto de treinta de estas reglas, incluyendo otras que implican la aplicación de *divide y vencerás*, la implementación *del tiempo compartido*, y otras. Muchas de estas reglas se corresponden con lo que se ha dado por llamar *métodos débiles*: métodos de resolución de problemas abstractos e independientes del dominio. La aplicación de una regla transforma el plan original en otro apoyándose en un método débil como aproximaciones sucesivas o divide y vencerás. Esto sugiere que las reglas de transformación de planes: sirven para aislar la estrategia de resolución de problemas subyacente empleada en un plan, de los métodos particulares utilizados para levar a cabo las submetas implicadas en la tarea. Esto permite que el planificador adapte un plan mediante la alteración de la estrategia de resolución en la que se apoya, a la vez que retiene, en la medida de lo posible, los métodos específicos que utiliza. En contraste, la técnica para aislar y reemplazar los pasos del plan deja la estrategia de resolución intacta, mientras sustituye los métodos específicos. Ambos métodos deben ser componentes de una teoría general de adaptación de planes.

5.2 Hammond

El conocimiento experto es cuestión de dos tipos de niveles de información del dominio: conocimiento de los problemas típicos que suelen ocurrir y conocimiento de optimizaciones particulares que se demostraron útiles en el pasado. Asociado con este tipo de conocimiento de problemas y oportunidades está la modificación de planes que evitan el primero y explotan el último.

Estas modificaciones pueden realmente aprenderse de un conjunto más primitivo de modificaciones y reparaciones. En particular, se aprenden cuando fallan las expectativas del planificador. En el caso de aprender a anticipar y evitar problemas, estos fallos en las expectativas se corresponden con los errores en la planificación. En el caso de aprender optimizaciones específicas, se corresponden con episodios del planificador reconociendo y explotando oportunidades.

CHEF utiliza conocimiento de errores pasados para evitar cometer el mismo error en un

nuevo contexto. Para evitar fallos, CHEF tiene que ser capar de anticiparlos, de darse cuenta de que van a ocurrir para darle al RETRIEVER la meta de evitarlo. Este es el papel del ANTICIPATOR, que tiene una base de conocimiento de fallos del CHEF.

La idea del ANTICIPATOR es la de predecir fallos sobre la base de características superficiales de una situación que causó fallos similares en el pasado. (si mezclas esto y lo otro entonces se estropea) Esta predicción hay que anticiparla (para despúes encontrar un plan que la resuelva). Por lo tanto, la memoria de CHEF se organiza en forma de una red de nodos, de forma que se enlazan fallos pasados con las características de las metas que las predicen.

Una vez que CHEF recupera un plan pasado, lo tiene que modificar para satisfacer cualquier meta que no cumpla todavía. Para hacer esto, CHEF posee una tabla de modificaciones típicas que indican como añadir nuevas metas a los planes existentes. Estas modificaciones son sustituciones, eliminaciones, concatenaciones, y mezclas de acciones y argumentos del plan.

Sin embargo, cuando se preveen fallos, CHEF necesita hacer cambios que van más allá de las modificaciones independientes del dominio sugeridas. Estos son casos donde la experiencia pasada ha demostrado que las modificaciones estándar llevaron a fallos, y por lo tanto se sugieren cambios específicos del dominio.

A la vez que anticipa el problema, también recupera la reparación que utilizó para resolver el problema en el caso anterior.

5.3 Hinrichs

Los procedimientos de adaptación más efectivos tienen tres consideraciones en cuenta:

1. La adaptación debe incluir el aprendizaje mediante la reutilización de las inferencias.
2. Debe ser independiente del dominio.
3. El conocimiento que se utilice para adaptar los casos debe estar también disponible para otros propósitos.[5]

La primera consideración se consigue normalmente agrupando inferencias en estrategias y desarrollando un vocabulario de dichas estrategias. En el vocabulario las estrategias son

[5] De la misma opinión es Callan en [CAL91].

independientes del dominio(2) y pueden utilizarse tanto para adaptar los casos y recuperarlos de las decisiones incorrectas(3).

Las funciones de la adaptación son:

1. Facilitar la toma de decisiones en problemas infra-restrictivos (under-constrained, es decir, generan demasiadas soluciones porque no se han especificado un número suficiente de restricciones) permitiendo que se reutilice un plan 'casi correcto' más que resolverlo desde el principio.
2. Resolver problemas sobre-rectrictivos mediante alternativas. En estos caso no hay solución por lo que el sistema cae en un estado inconsistente o posee una solución incompleta y no puede continuar.
3. Relajar las restricciones dándoles una prioridad (según importancia o prioridad) y relajando la menos importante.
4. Amplia el dominio del vocabulario diseñando nuevos componentes como variaciones de otros conocidos. Por ejemplo, crea situaciones nuevas eliminando características secundarias que violen alguna restricción.

Para alcanzar estos objetivos se utilizan las siguientes estrategias

Búsqueda local. Cuando un valor viola una restricción, se suele encontrar una alternativa aceptable en las cercanías del espacio de búsqueda. En lugar de eliminar el valor y buscar otro nuevo, se intenta buscar una especialización y una generalización del valor que lo satisfaga. Dos beneficios:

La jerarquía semántica proporciona un espacio de búsqueda limitado y bien definido en ayuda del espacio de resolución del problema que suele ser ilimitado y mal definido.
Si muchas decisiones subsiguientes dependen del valor previo, puede que no necesiten retractarse si el nuevo valor es suficientemente similar.

Compartición de Funciones. Cuando un problema es relativamente nuevo, la experiencia podría sugerir una solución cuya estructura es casi, aunque no lo suficiente, adecuada. En este caso, el sistema de resolución de problemas debe adaptar no sólo los valores específicos, sino también la estructura del problema. La compartición de funciones es una estrategia que es aplicable cuando las consideraciones económicas o estéticas recomiendan la utilización de un único mecanismo o acción para servir diferentes funciones. Modifica la estructura de un problema combinando aquellas variables que sean funcionalmente equivalentes. La compartición de funciones se implementó en JULIA como un requisito que relaciona dos slots o escenas tal que se elimina una si su función por defecto está incluida en la otra.

División de Funciones. Cuando varias restricciones son conflictivas entre si, a veces es posible dividirlas y resolverlas independientemente. La división de funciones incrementa el número de variables en un problema para simplificar la satisfacción de requisitos.

Asociado con esta estrategia está el criterio para determinar cuando se puede dividir una variable. Estos criterios caen en tres categorías.

- **Naturaleza de la variable**. La variable debe representar un conjunto de elementos se forma que se pueda dividir en subgrupos.
- **Fuente de restricciones conflictivas.** Las restricciones conflictivas deben derivar de una característica común, tal como las características de un episodio.
- **Independencia de las restricciones.** Debe ser posible dividir las restricciones por fuente de forma que cada partición pueda satisfacerse.

Cuando se verifican las restricciones, la variable puede dividirse en dos nuevas variables que son entonces re-restringidas y resueltas independientemente.

Transformación. La búsqueda local puede fallar en los problemas que sean menos rutinarios. Entonces se puede tomar el valor inconsistente y transformarlo con la adición, eliminación, o sustitución de componentes.

5.4 Alex Kass

Construye un sistema de Explicaciones Basadas en la Adaptación. Cuando un sistema necesita una adaptación, el sistema construye una a partir de una similar de un caso pasado, adaptándola al nuevo contexto.

La base de casos se compone de explicaciones (XPs) que son las que adaptará el sistema. Las XPs son redes de inferencia que parten de las anomalías que explican hasta las posibles causas.

Las explicaciones pueden fallar de diferentes formas, y existen múltiples formas de arreglar la mayoría de los fallos. Una estrategia para adaptar un XP es un algoritmo de búsqueda de la base de conocimiento para encontrar substituciones, generalizaciones, o especificaciones que son necesarias para arreglar un tipo particular de fallo.

Los dos tipos de fallos principales son:

fallos de plausibilidad: se corresponden con explicaciones que no tienen sentido porque contradicen algunos aspectos del modelo del mundo (de la situación actual). Es decir, a pesar de que comparten ciertas similaridades superficiales, no son aplicables porque no tienen nada que ver (las dos situaciones).

fallos de vaguedad: se corresponden con explicaciones que no están suficientemente detalladas, no son suficientemente convincentes, o no contienen el tipo de información que se ajusta a las necesidades del sistema de explicación. Faltaría información.

El corazón del proceso de adaptación es una librería de estrategias, cada una de las cuales es un programa que es capaz de arreglar un tipo de fallos en la explicación. Para cada uno de los fallos en la explicación que controle el tweaker (el módulo que adapta las explicaciones) existirá un conjunto de estrategias aplicables. el tweaker tasa este conjunto según haya sido la eficiencia de la estrategia en el pasado, de lo a menudo que haya funcionado, y en qué medida estaría relacionada con el fallo a mano, y ejecuta las estrategias en orden mejor el primero hasta que una produce una explicación aceptable. Algunas de las estrategias son las siguientes:

Sustituciones. Las sustituciones intentan arreglar problemas de plausibilidad mediante la sustitución del contenido de un slot en una de las creencias del XP errado con un componente que tiene las mismas consecuencias causales (se sustituye uno de los slots subiendo por la jerarquía del mismo, hasta encontrar las categorías a las que pertenece, y recorrer los enlaces para encontrar las acciones asociadas con dichas categorías).

Normalmente, el tweaker se define por tres cosas: la parte del XP que modifica, el método de búsqueda del conocimiento del domino necesario para hacer la modificación, y un conjunto de tests que deben ejecutarse sobre la modificación para comprobar si tiene sentido. Algunas sustituciones posibles son:

Reemplazar una vieja acción con una acción relacionada de cerca en la jerarquía de acciones.
Reemplazar una vieja acción con una acción indexada como causante de uno de los eventos en el XP.
Sustituir un viejo actor por uno que se sabe que tiene motivaciones nombradas en el XP
Sustituir un viejo actor con un actor relacionado con el viejo actor que pudiera haber realizado la acción.

Generalizaciones. Las generalizaciones arreglan fallos en la plausibilidad y fallos en la

aplicación produciendo una versión de una aplicación que es aplicable a una mayor gama de situaciones, con el coste de eliminar algunos de los detalles de la hipótesis. Esto se puede realizar de dos formas:

Generalizando componentes mediante la alteración del contenido de algún slot en una de las creeencias del XP. Se diferencian de las sustituciones en que ellas intentan generalizar los componentes no válidos en lugar de buscar un alternativa. Ejemplos:

- Generalizar viejas acciones para hacerlas compatibles con el nuevo actor.
- Generalizar las restricciones en un actor para hacerlo compatible con el actor actual.

XPs simplificadores generalizan la estructura del xp haciendo cambios más globales que los que simplemente alteran uno de los componentes. (como eliminar creencias problemáticas.

Especificaciones. Las especificaciones arreglan los problemas en la vaguedad encontrando detalles a añadir a una explicación, haciéndola menos general, aunque más rica en contenido en información. Existen dos categorías:

Especificaciones de componentes que añaden detalle a la explicación haciendo más específica la descripción del valor de algún slot. Ejemplos:

- Especificar la descripción del actor encontrando un actor que reconozca la descripción que se nombra en algún lugar del XP o en la historia que se está procesando.
- Especificar la descripción de un actor encontrando un actor que reconozca la descripción que tiene las motivaciones descritas en el XP.

XP elaboradores añaden detalle a una explicación construyendo más estructura. Estas estrategias hacen cosas como añadir enlaces causales entre creencias en la explicación, o añadir explicaciones adicionales que apoyen alguna de las creencias, o una dos XP para formar una explicación más completa. Ejemplos:

- Añadir a las conexiones causales entre dos creencias fusionándolas en otra XP que explique como una puede causar la otra.
- Explicar una decisión hecha con uno de los actores con la fusión en un explicación de porque serían más importantes los buenos efectos de la decisión para el actor que para la mayoría.
- Explicar una decisión hecha con uno de los actores mediante la fusión en una explicación de porque son menos importantes los malos efectos de la decisión para el actor que para la mayoría.

- Explicar una decisión hecha por uno de los actores con la fusión en una explicación de porque el actor podría no haber conocido algún mal efecto de la decisión.

Un tema crucial en las teorías de adaptación es el de a que nivel de conocimiento pertenece en el proceso de adaptación. Algunos piensan que muy generales (pocas estrategias independientes del domino, siendo poco eficientes), otros prefieren las adaptaciones específicas del dominio (necesitan muchas). Las descritas en esta última aproximación son intermedias.

Capítulo II

Especificación de la Arquitectura Funcional propuesta

En este capítulo identificamos las principales lagunas que posee el área a partir del estudio detallado en el Capítulo anterior. Estas conclusiones serán las que nos permitan especificar la propuesta de una nueva arquitectura de un Sistema Basado en el Conocimiento, aunque centrándonos únicamente en su especificación formal. Las principales decisiones de diseño serán justificadas en un capítulo posterior. La definición del Sistema se realizará de manera incremental, empezando por los componentes primitivos, y construyendo sobre estos las definiciones más complejas.

Como cualquier herramienta de desarrollo software, el Sistema se compone básicamente de un esquema de representación y de un mecanismo de control que actúa sobre él. Nosotros proponemos un esquema híbrido de representación del conocimiento compuesto por una memoria episódica o base de casos, de un modelo conceptual jerárquico, y por un modelo causal del dominio. De esta manera, somos capaces de descomponer y trabajar independientemente con las distintas naturalezas de una unidad de información. El mecanismo de control define los procedimientos que hacen que el sistema alcance una solución. Estos procedimientos consisten básicamente en la aplicación de una cierta función de similaridad sobre cada uno de los casos de la base para determinar el grado de afinidad de cada uno de ellos con la situación actual. Esta función de similaridad tendrá en cuenta la situación contextual de cada unidad de información del caso actual, accediendo a los distintos elementos representacionales del Sistema para definir dicho contexto.

1 Análisis crítico y propuesta de investigación

En esta sección trataremos de analizar los principales problemas que poseen las distintas arquitecturas comentandas en el capítulo anterior desde un punto de vista globalizador, buscando deficiencias o lagunas que afecten al campo de los CBR en general, para a partir de ellos proponer un posible diseño alternativo de arquitectura. Seguiremos para esta discusión la cronología seguida en el capítulo I a la hora de discutir los distintos tópicos (representación, recuperación, etc..), aunque primero comentaremos algunas conclusiones que consideramos importantes y que afectan a la globalidad del campo CBR.

Fata de fundamentación Cognitiva

Los CBR, a diferencia de cualquier otro paradigma que se haya estudiado en el campo de la Inteligencia Artificial, tratan de abordar procesos mentales complejos y a un nivel superior de abstracción a los otros modelos de resolución de problemas usualmente utilizados. Nos referimos al pensamiento analógico. Éste entendido, no tanto en el sentido interdominio (una experiencia en un dominio nos recuerda otra en otro dominio que no comparte aparentemente ninguna característica 'superficial') sino en el nivel episódico dentro de un mismo dominio. La mayoría de los trabajos revisados están relacionados con la analogía intradominio, aunque algunos autores se han atrevido a la investigación de analogías semánticamente distantes [HOF95]. Los principales problemas que surgen con este último tipo de analogía están relacionados con los algoritmos de recuperación, al ser sólo posibles con grandes bases de conocimiento multi-dominios en las que el tiempo necesario para hacer las conexiones analógicas es siempre prohibitivo [WOL95].

La especificación de cualquier modelo computacional que trate de emular el razonamiento analógico debe venir fundamentada en alguna teoría cognitiva que guíe y valide los resultados experimentales. Gaines [GAI89] propuso una modelización de los distintos tipos de conocimiento que intervienen en la resolución de tareas expertas, la cual se ajusta perfectamente a la problemática CBR. Identifica y distribuye los tipos de conocimiento en una jerarquía en la que los niveles superiores se identifican con las estructuras de conocimiento más complejas y los inferiores con las más simples. Establece los mecanismos de comunicación que deben existir entre cada uno de los niveles de forma que se pueda producir la adecuada transferencia de información. Un cierto nivel de la jerarquía es capaz de resolver problemas más complejos que los que se pueden resolver en los niveles inferiores, utilizando

para ello los recursos que su propia definición le proporciona, más los específicos de los niveles inferiores. La propuesta de Gaines (figura 6) sitúa a los sistemas CBR en el nivel 3 o computacional, dejando a los sistemas de producción en el nivel inferior, y a la analogía interdominios en el superior.

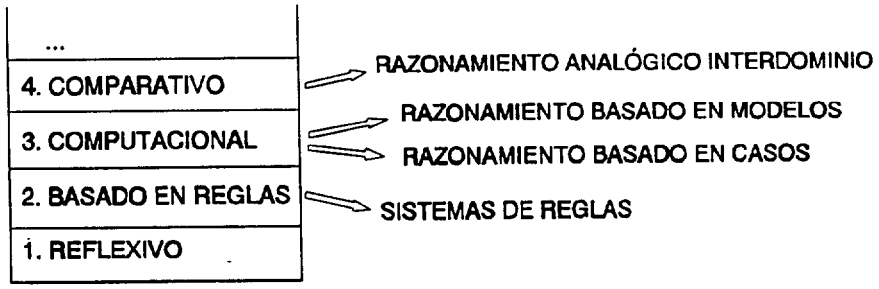


Figura 6

Sin embargo, casi ningún diseño CBR de los analizados tiene una fundamentamentación cognitiva suficientemente consistente. Y esta falta de base apunta como una de las principales razones de sus limitaciones, dando lugar a arquitecturas complicadas artificiosamente. Por ejemplo, a partir de la escala de conocimiento de Gaines se deduce que no es posible desarrollar un sistema a un nivel superior si no se abordan las tareas de los niveles inferiores. Esto se traduce en la coordinación del sistema CBR (nivel 3) con un conocimiento causal del dominio (nivel 2). Trabajos como el de Goodman [GOO89], la herramienta de desarrollo de aplicaciones CBR *CBR-Express* y otros han no han tenido en cuenta este hecho y ello ha conducido a sistemas extremadamente ingenuos. No sólo eso, la jerarquía de Gaines también nos indica que el nivel superior debe tener la iniciativa durante los procesos de resolución, y en ningún caso a la inversa. En base a esta afirmación, carecería de sentido el desarrollo de un sistema CBR de apoyo a un mecanismo de razonamiento basado en reglas como han desarrollado Koton con CASEY, o Golding [GOL91]. Parece más razonable la postura contraria, que el conocimiento causal proporcione justificaciones, o explicaciones sobre determinadas decisiones del razonamiento. Así lo hacen sistemas como CHEF de Hammond [HAM91]. Tampoco suscribimos dotar de la misma entidad a ambas arquitecturas (reglas y cbr) como en el trabajo de Rissland [RIS91]. Una cosa es que ambas estructuras cooperen y se transmitan información de cara a obtener un funcionamiento más eficaz, y otra es permitir que sea la estructura más simple (ej. regla) la que decida sobre la oportunidad y calidad de la información que otra estructura más compleja (ej. cbr) le ofrece. Existen ciertos aspectos del conocimiento que no pueden ser asimilados por la estructura de menor complejidad, lo que conduciría a una pérdida de parte de la potencialidad de la de mayor riqueza.

Carencia de criterios de rendimiento

Otro hecho significativo que resalta de este estudio es la gran variedad de propuestas existentes en las que es difícil (por no decir imposible) encontrar un nexo u origen que las relacione de alguna manera. Además, cada propuesta es normalmente defendida mediante una implementación sobre un dominio de conveniencia, y que usualmente es distinto de cualquier otro dominio utilizado en otras aplicaciones. No pretendemos indicar que esto se realiza con alguna mala idea para dificultar posibles tareas de validación y contraste. Sería muy difícil aprovechar una base de conocimiento desarrollada para un sistema sobre otra arquitectura, debido a lo diferentes que pueden llegar a ser dichas arquitecturas y a los distintos requisitos que imponen sobre el dominio que pretenden modelar. Y mucho más difícil si no se cuenta con la asistencia de un experto que permita 'refinar' la transferencia de información de una base a la otra. Este es un aspecto y un inconveniente muy importante cuando se pretende validar y contrastar un trabajo investigador. Como apuntamos anteriormente, muy pocas propuestas tienen alguna fundamentación cognitiva, y esa será posiblemente, junto con la complejidad de la tareas que pretenden resolver, la causa de la divergencia de todas ellas. Tal vez podramos citar un origen común en aquellos trabajos que a nuestro entender son más profundos y están mejor fundamentados; este es Schank y su trabajo *Dinamic Memory. A theory of reminding in computers and people* [SCH82], lo cual es consistente con el trabajo comentado anteriormente de Gaines. Hoy por hoy, y a excepción de algunas pocas colaboraciones, parece que esta divergencia y esta disparidad de trabajos va a continuar, de forma que será muy difícil que, en a medio plazo, se alcance un cierto consenso sobre lo que puede ser una normalización u homogeneización de los componentes y funcionamiento de los sistemas CBR.

Coordinación con las metodología de desarrollo de SBC

Finalmente, es necesario hacer constar un factor muy importante que si bien estaba en el origen del desarrollo de estos sistemas, se ha ido marginando hasta que ya sólo se nombra en los artículos de promoción o introducción de esta tecnología. Nos referimos al proceso de adquisición del conocimiento previo al funcionamiento de cualquier sistema experto. En origen, uno de los argumentos que se daban a favor de la tecnología CBR era que minimizaba el tradicional cuello de botella que representaba la adquisición del conocimiento, ya que 'sólo' era necesario introducir casos en el sistema y el resto lo hacía el motor del sistema. Pero el campo ha evolucionado, y hoy se desarrollan sistemas con unas exigencias altísimas a la hora de identificar, adquirir, y codificar múltiples aspectos del conocimiento, que deben ocupar su propio lugar dentro de la arquitectura en cuestión (ver por ejemplo la propuesta de Blau [BLA91] con eventos, acciones, estados, interpretaciones, casos, planes, y conceptos; y cada uno con sus correspondientes enlaces). Lejos de ser paradógico, este comportamiento es cada vez más usual en los sistemas que se están desarrollando. Sin darnos cuenta estamos

desarrollando sistemas tan complejos que serán inviables, no porque no sean capaces de abordar eficientemente la tarea que se les asigna, sino porque no se han desarrollado paralelamente las técnicas y las herramientas de Adquisición del Conocimiento y de formalización necesarias para poner en marcha una aplicación de este tipo. Pensamos que las nuevas arquitecturas que se propongan deben tener en cuenta estos hechos; es necesario ajustarse a las restricciones que impone el estado actual de la adquisición del conocimiento a partir de un experto humano. En el mejor de los casos, es necesaria una primera inicialización de la base de conocimiento de forma que el sistema pueda evolucionar posteriormente; pero no hay que olvidar que los algoritmos de 'aprendizaje' no hacen más que reorganizar el conocimiento que actualmente posee el sistema. Aún así, se hace necesaria una evolución de los métodos tradicionales de desarrollo de SBC, de forma que sean capaces de abordar los nuevos problemas que esta nueva tecnología les impone. La simple selección de un buen conjunto de situaciones a incluir en el sistema no es una tarea sencilla, y esa es una responsabilidad que descansa hoy por hoy en el equipo formado por el Experto humano y el Ingeniero del Conocimiento.

Después de realizar esta crítica general al área, tenemos que comprometernos en el desarrollo de una propuesta que subsane los defectos anteriormente citados. A partir de este momento vamos a realizar una breve revisión de cada uno de los tópicos analizados, los cuales servirán de introducción de las características más sobresalientes de la propuesta realizada.

Como en cualquier aplicación de IA, la representación fija muchas de las decisiones de diseño sobre el resto de la arquitectura, como la recuperación de casos, o la valoración de la similaridad, y por consiguiente el rendimiento global del sistema. Por eso, debe ser objeto de un especial análisis a la hora de valorar las aportaciones realizadas hasta la fecha. En línea con lo comentado anteriormente, no nos parecen acertadas las líneas que afrontan el problema apoyándose exclusivamente en una arquitectura CBR. No obstante son muy pocos los que han optado por esa alternativa; la mayoría utilizan un esquema combinado ya sea con algún espacio paralelo de conceptos, con un modelo parcial, o incluso con un sistema cooperante del tipo Model-based. Consideramos esta alternativa es, en principio válida, ya que permite seguir las directrices de Schank en cuanto a los procesos de razonamiento analógico. Además una arquitectura híbrida presenta una serie de ventajas adicionales como:

- La posibilidad de aprovechar el oportunismo haciendo recaer parte de la responsabilidad sobre una estructura u otra
- La complementariedad de las estructuras implicadas minimizarán el efecto que pueda tener una deficiencia en algún aspecto de dichas estructuras.

- Se produce un ahorro cognitivo a la hora de modelizar el dominio.

Si bien estas ventajas pueden haber estado en la intención de muchos de los autores, no han afrontado el problema desde el ángulo correcto. O integran excesivamente distintos tipos de conocimiento, o permiten demasiada iniciativa al paradigma de menor entidad. Dentro del primer grupo tenemos trabajos que descomponen casos en una jerarquía conceptual, trabajos que establecen una relación directa entre reglas y casos concretos, o distribuyen los casos en un grafo con una semántica causal principalmente (XPs). Consideramos que una separación clara de cada formalismo representacional posee una serie de ventajas como son:

- Es posible definir los cometidos de cada componente.
- Las estructuras resultantes son mucho más simples y sencillas de gestionar, lo cual beneficia especialmente cualquier esfuerzo de desarrollo.
- El proceso de adquisición del conocimiento puede ser mucho más estructurado, facilitándose el proceso de transferencia de la información.
- La consistencia de cada estructura puede ser evaluada más eficientemente. Y si se trata de una de las estructuras típicas de IA, seguramente existirán estudios sobre este aspecto que puedan aplicarse.
- La validación de completitud de cada mecanismo puede relajarse, siempre y cuando se permitan modelos incompletos, y existan medidas de validación del sistema completo
- Pueden diseñarse algoritmos de coordinación y control de las estructuras más modulares, y que no precisan del acceso a partes de las estructuras que no sean necesarios.

La integración de múltiples estructuras exige de la especificación de los mecanismos de control, y de comunicación entre ambas; la forma en que los mecanismos se complementarán y bajo que condiciones; que tipo y en que formato se podrán pasar información; a que partes de qué estructura tendrá acceso cada una de ellas, etc..

La arquitectura propuesta opta por la coordinación de tres estructuras: una base de casos, un modelo causal, y una jerarquía de conceptos. Con esta propuesta de representación es posible diseñar un sistema que subsane las carencias anteriormente citadas. Partiendo de la afirmación realizada por Long [LON94] en el sentido de que una buena analogía es reflejo de un argumento causal subyacente, nos decantamos por la inclusión de un modelo causal porque, además, lo consideramos imprescindible si perseguimos ser coherente con los trabajos de Schank y Carbonell. Es decir, permitir no solo reutilizar una solución de un caso previamente resuelto, sino el camino que se llevó para su resolución. Además, pretendemos mejorar las deficiencias del trabajo de Carbonell permitiendo la reutilización tanto de segmentos de la resolución, como de componentes de la solución. Existen estudios que

permiten la validación de completitud y consistencia sobre sistemas de producciones [PER89] que, dependiendo de lo restrictiva que sea la gramática del modelo, puede que sean aplicables en este contexto. También hacemos uso de una jerarquía de conceptos se facilitar el razonamiento analógico, y porque se ha demostrado suficientemente como una potente estructura representacional, tanto en arquitecturas CBR, como de IA en general. Además, como se especifica en [YAN93] la construcción de una jerarquía de conceptos puede automatizarse, puede realizarse un proceso de inferencia mediante una ajuste parcial, y existen múltiples técnicas desarrolladas para realizar agrupamientos conceptuales. Todo esto sin perder de vista que es el sistema CBR el que debe llevar la iniciativa y el control de todo el proceso de resolución.

Esperamos que un sistema con las características como el descrito permita representar un amplio segmento del conocimiento en el dominio de la aplicación, con un menor esfuerzo que sistemas equivalente, tanto de ingeniería como representacional, gracias al grado en que las estructuras se pueden complementar y a la posibilidad de definir y trabajar con modelos sólo parcialmente descritos.

En relación con la recuperación, los métodos más complejos son aquellos que trabajan sobre una estructura de casos distribuida jerárquicamente, donde como se comentó anteriormente, cada nodo contiene tanto información estructural como causal. Esto conlleva dos efectos en lo que al algoritmo de recuperación se refiere: primero su complejidad es mayor al tener que trabajar sobre estructuras de conocimiento más complejas, teniendo el propio algoritmo que separar las distintas naturalezas de la información que se mantiene en dichas estructuras. Y en segundo lugar, aunque son algoritmos que son paralelizables, puede que esa sea la única opción si se quiere hacer funcionar sobre una base de casos suficientemente grande.

Por otro lado tenemos a los autores que como Kopeikina [KOP88] consideran que el método de recuperación viene condicionado directamente por el dominio, lo cual hace a su método poco aprovechable. En todo caso, un método de recuperación debe venir condicionado por la tarea que se espera que resuelva el sistema, pero no por la aplicación concreta que se desarrolla.

Y finalmente están los métodos dinámicos, los cuales se apoyan en una representación causal del sistema que justifique una determinada reestructuración de la base de conocimiento para facilitar la recuperación más eficiente en el futuro. Un esquema dinámico de recuperación no persigue realmente otra cosa que asimilar nuevas situaciones sin tener que almacenar un nuevo caso completamente. Pensamos que antes de considerar formas de rehacer una base de conocimiento, sería necesario demostrar que las estructuras de representación son

las adecuadas para la tarea que deben cumplir, y no a la inversa, definir una estructura representacional en base a las posibilidades que ofrece para su auto-reestructuración.

Tratamos de evitar la polémica sobre si es mejor utilizar características de bajo nivel o de alto nivel (elaboradas) durante la recuperación. Es evidente que ambos tipos de características deben participar en el proceso de recuperación. Además, un proceso de recuperación debe utilizar toda la información disponible, y el problema no está tanto en decidir si utilizar o no características de alto nivel, sino en como calcular la aportación de cada una. Asignar 'pesos' que indiquen la importancia de cada atributo definido en el sistema es una tarea altamente subjetiva, y que da lugar continuamente a inconsistencias en la base que no son fácilmente detectables. Sea la asignación de pesos 'manual' mediante una coordinación entre el experto en el dominio y el ingeniero del conocimiento, o 'automática' mediante algoritmos de inducción, no es, ni desde nuestro punto de vista, será una forma eficiente de asignar a cada unidad de conocimiento su peso específico exacto. Además, el problema no está sólo en el método de ponderación, sino en cómo hacer que ese 'factor' de importancia dependa del contexto en el que se desarrolla un episodio u otro. No pesa lo mismo un atributo para una situación que en otra. La solución debe pasar, nuevamente, por la definición de unas relaciones entre distintas partes del conocimiento que premien, en función de cada contexto, a cada unidad de información en su justa medida. Por ejemplo, un concepto será más importante, cuanto más relacionado esté, en determinado momento, con el resto de las estructuras de conocimiento.

Esta función debe venir facilitada por los enlaces adecuados entre las distintas partes del conocimiento representado. La indexación será la encargada de proporcionar, en cada instante el tipo de enlace necesario, siempre que este exista. Entonces, ponderar la importancia de un tipo de enlace, sí que puede ser una tarea mucho más asequible (principalmente por el reducido número que existe) que ponderar cada uno de los atributos del sistema. Sin embargo, ningún autor tiene esta visión de la indexación. Usualmente, la indexación se utilizaba para 'extraer' determinados aspectos de la información que estaban empaquetados en una estructura muy compleja, como podría ser una relación causal dentro de un XP. Desde el momento en que se separan los distintos tipos de conocimiento en estructuras claramente diferenciadas, estos tipos de índices dejan de tener sentido.

A la hora de medir la similaridad, usualmente se dispone de dos métodos: utilizar un test booleano, o una función de scoring. El primero consiste en aplicar un test, que tradicionalmente consistía en comprobar una semejanza causal entre el caso de entrada y los de la base. Si se recuperaban demasiados, entonces era necesario completar la descripción del caso actual de alguna manera. El otro método consiste en calcular un valor de similaridad para cada caso en memoria. Este valor se calcula mediante una suma de las aportaciones de cada característica que tienen en común las situaciones de entrada y de memoria. La debilidad

de ambos métodos es notable. El primero porque usualmente se recuperaban demasiados casos, y se hacía necesario realizar test adicionales que filtraran o seleccionaran la lista recuperada. El segundo, porque se apoyaba en un sistema de asignación de pesos débil en origen, como se comentó anteriormente. Además, el sistema de scoring es mas coherente desde el momento en que parece razonable reconocer que existen casos que son más parecidos que otros. De hecho, algunos autores combinan la primera aproximación con alguna heurística que permita 'afinar' el grado de similaridad entre dos situaciones.

En general, el proceso de recuperación carece de un sistema de validación de la solución recuperada. En el mejor de los casos esta validación ser realiza mediante la ejecución de una serie de pruebas almacenadas en el caso recuperado, o comprobar la consistencia del modelo causal del caso recuperado con los datos actuales. Pero esta información no siempre está disponible, o se utiliza para discriminar entre casos similares, por lo que deja de ser válida en esas situaciones. Aunque parezca extraño, tampoco hay muchos sistemas que se arriesguen a proporcionar múltiples soluciones a una entrada (aún cuando pudiera tener sentido, como en las tareas de diagnosis o interpretación). El problema es el mismo: ser capaz de definir el estado en el que se encuentra la recuperación en un momento determinado.

Nosotros consideramos que la definición de una función de similaridad en base a la ponderación de una serie de índices más que sobre pesos asignados a las características facilita el proceso de recuperación, permitiendo la definición de una serie de medidas para la evaluación del estado en el que se encuentra la recuperación y la validación del caso o los casos que se proponen para su recuperación. Desviar el centro de atención desde el *atributo* hacia el *índice* es esencial si queremos aprovechar dinámicamente la información que proporciona cada una de las estructuras representacionales disponibles. La función de similaridad es, de esta forma, capaz de recoger en cada momento el entorno que rodea a cada una de las unidades de información que describen a la situación actual. Sólo así será posible valorar la aportación de cada unidad de información en su justa medida. Además, para el ajuste de esta función se podrá utilizar un método por aproximaciones por gradiente, ya que podemos suponer que existe una cierta relación entre los distintos tipos de índices que consideramos para la valoración de la similaridad y por lo tanto es factible encontrar un máximo local dada una aproximación inicial, evitando así los inconvenientes de los algoritmos tradicionales de inducción.

En cuestión de adaptación de casos, podríamos distinguir dos grandes grupos: por un lado las adaptaciones estratégicas, que son independientes del dominio. Y por otro las tablas de adaptaciones que sí dependen de cada aplicación en concreto. Las estratégicas son insuficientes, por lo que normalmente van acompañadas de conocimiento específico del dominio sobre el que se aplica el sistema. Las tablas de adaptaciones tienen el inconveniente

de tener que especificar todas las transformaciones posibles para cada situación, lo cual puede ser una tarea inviable. Las estratégicas proporcionan un nivel de abstracción mayor, de forma que cada tipo de adaptación es capaz de cubrir múltiples situaciones. Al igual que ocurre con la capacidad de evolución, la definición de una estrategia concreta de adaptación no será posible hasta que el resto de la arquitectura del sistema haya demostrado su funcionalidad. Sin embargo, deben especificarse las bases de un posible modelo de adaptación. Así, consideramos que las operaciones estratégicas básicas de sustitución, eliminación, y adición pueden llevarse a cabo siempre y cuando el modelo causal tenga una capacidad de inferencia plena. Esto, junto con la capacidad de especificar para cada meta las posibles repercusiones mediante un lenguaje procedural simplificado a las expresiones usuales permitiría completar unos mecanismos potentes de adaptación.

Un último aspecto que consideramos de crucial importancia es la Verificación y Validación de los sistemas CBR. A pesar de que reconocemos la relevancia de estas cuestiones no le hemos destinado ninguna sección porque no existen apenas trabajos representativos, y se podría decir que es una de las principales lagunas que posee esta área. Cuesta pensar que con la cantidad de sistemas distintos que existen no se haya destinado al menos una pequeña parte del esfuerzo a validar el funcionamiento del sistema que se diseña de una forma más o menos formal. Además, en los pocos trabajos revisados bajo este epígrafe no se tiene una idea clara del significado de estas tareas, o se atacan desde un punto de vista incompleto. Así, en [BARE89b] se llama 'evaluación' (vocablo que usualmente abarca los procesos de verificación y validación) al cálculo del rendimiento del sistema frente a distintas condiciones de entrada (lo que en el mejor de los casos podría ser un proceso de validación). De igual forma, Santamaria [SAN94] estudia las variaciones en el rendimiento frente a cambios en los parámetros del sistema, lo cual no se puede incluir dentro de lo que es un proceso de verificación. Y Francis y Ram [FRA94] proponen una metodología que sólo es capaz de afrontar uno de los muchos problemas asociados con la verificación: el *'utility problem'*, o la degradación de la base de casos provocada por una asimilación excesiva de situaciones resueltas. Un proceso de verificación debe evaluar el estado interno del sistema, identificando qué problemas hay, y donde se encuentran. La validación sí es un proceso más genérico y que puede ser abordado con las técnicas que usualmente se usan en IA, como el test de Turing, validación confrontada, etc.[6] Únicamente en [COH89] se proporcionan unas directrices correctas sobre lo que debe ser un proceso de evaluación, aunque no se aborda ninguno para un sistema concreto. Algunas de ellas son:

- ¿Cuáles son las métricas para evaluar el método?
- ¿Sobre qué modelos se asienta el sistema? ¿están justificadas todas las decisiones de

[6] En [OKE93] y en [GUP91] se proporciona un amplio abanico de métodos de validación que pueden ser útiles a la hora de evaluar este tipo de sistemas.

diseño?

- ¿Aborda el sistema exactamente la tarea o segmento de tarea?
- ¿Por qué funciona el método (o no funciona)? ¿son inherentes las limitaciones del método, o simplemente no han sido todavía abordadas?

Nosotros pensamos que el propio sistema debe proporcionar los mecanismos para su verificación. La arquitectura que se plantee debe estar en condiciones de proporcionar un conjunto de medidas acerca del estado interno de sus estructuras. Creemos que un sistema como el que estamos empezando a describir posibilita dicho análisis, y esperamos poder demostrarlo a lo largo de la presente memoria.

2 Elementos Representacionales

Hemos agrupado los elementos representacionales en tres grupos en función de la estructura de alto nivel a la que pertenecen. Los elementos comunes (o que participan en las tres estructuras representacionales) también forman su propio conjunto de descripciones. De esta manera, organizamos los elementos que componen el esquema de representación atendiendo a si permiten describir la

- Unidad de Información
- Jerarquía Conceptual
- Red Causal
- Memoria de Casos

2.1 Unidad de Información

La Unidad de Información (UI) constituye el núcleo sobre el que se asientan el resto de elementos representacionales del Sistema. La UI se define como aquella porción de conocimiento que permite describir una característica del Dominio. En la terminología usual de desarrollo de software podría ser un objeto, o un concepto. No es el objetivo de este capítulo discutir que es un concepto y que no, tarea que no es tan sencilla como parece. Para eso están las diversas metodologías software como puede ser la orientada a objetos de Booch [BOO94]. Booch defiende que un objeto debe poseer un estado, un comportamiento, y una identidad. Y esos son los criterios que un ingeniero del software o del conocimiento podría tomar para decidir si un determinado elemento del dominio merece ser modelizado como un concepto.

Definición 2.1 Un concepto en el sistema viene definido por un conjunto de atributos que lo caracterizan. Los atributos son propiedades o cualidades que permitirán distinguir entre conceptos del dominio. Por ejemplo, podemos definir el concepto *paciente* en base a los atributos de *nombre, edad, sexo, ...* Cada atributo, a su vez, vendrá definido por una serie de características que lo definen, como puede ser el tipo del atributo. Un concepto puede venir definido por un número arbitrario de atributos, aunque siempre debe poseer al menos uno que es el atributo *nombre del concepto*, el cual será distinto para cada concepto definido en el dominio.

De esta forma, debemos definir en el Sistema dos conjuntos:

O: es el conjunto formado por todos los objetos (conceptos) del Sistema
A: es el conjunto formado por todos los atributos que definen cada elemento de O

De forma que existe una relación *h* entre O y A. Esta relación tiene la forma de

$$\forall o_i \in O,\ \exists P \subseteq A,\ h(o_i) = P$$

Para poder definir el conjunto A, también será preciso definir otro conjunto de características de los atributos (S) definiéndose una relación *i* similar a la anterior desde A a S:

$$\forall a_i \in A,\ \exists T \subseteq S,\ i(a_i) = T$$

A diferencia de los conjuntos anteriores, esta vez el conjunto S es un conjunto cerrado (con un número finito de elementos), los cuales pasamos a definir.

Cada atributo en el Sistema, debe venir definido por:

Identificador. Consiste en una cadena de caracteres que permite identificar unívocamente al atributo.

Tipo. El cual puede tomar uno de los siguientes valores:
Numérico. (ej. edad = 29)
Cadena de Texto. (ej. nombre = Antonio Pérez)
Lista de Valores Legales. (ej. color = Rojo, o Amarillo, o Azul)

Fórmula. Valor a calcular mediante alguna expresión. (ej. trienios = años / 3) siguiendo la gramática BNF usual para expresiones:

E ::= E + T | E - T | T
T ::= T * F | T / F | F
F ::= (E) | constante | identificador

E = expresión; constante es una constante numérica o string, según corresponda; y identificador es la pareja (objeto, atributo).

Además, en función del tipo del atributo será necesario definir las siguientes características adicionales:

Límite Inferior. Para el tipo numérico.
Límite Superior. Para el tipo numérico
Valores Legales. Para el tipo Valores Legales

Ahora ya podemos definir con precisión lo que es una Unidad de Información.

Definición 2.2 Una UI es cada elemento $o_j \in O$, $o_j = \{a_1, a_2, ..., a_n \mid a_i \in A, 1 \leq n \leq |A| \}$, donde uno de los a_i debe ser el atributo por el que se puede hacer referencia al concepto (nombre).

Ejemplo 2.1 Podríamos definir el objeto *animal* de la siguiente manera:

atributo	definición del atributo
nombre-del-objeto	identificador: nombre-del-objeto tipo: cadena de texto
nombre	identificador: nombre tipo: cadena de texto
sexo	identificador: sexo tipo: lista de valores valores legales: macho, hembra
edad	identificador: edad tipo: numérico límite inferior: 0 límite superior: 25

Definición 2.3 Si O es el conjunto de todos los conceptos definidos en el Sistema, O debe verificar la propiedad de consistencia, por la cual ninguno de los elementos de O pueden compartir el identificador del concepto.

$\nexists\, o_i = \{a_1, a_2, ..., a_n \mid a_i \in A, 1 \leq n \leq |A| \}, o_i' = \{a_1', a_2', ..., a_n' \mid a_i' \in A, 1 \leq n \leq |A| \} \in O$, tal que $a_j = a_j'$, donde j es el índice del atributo que representa el identificador del concepto.

2.2 Modelo Conceptual

Definición 2.4 Sea O el conjunto no vacío de los conceptos del Dominio, y $P \subseteq OxO$. Entonces, una jerarquía conceptual *J* es el par (O, P), el cual es un grafo dirigido en O sin ciclos ni lazos, donde O es el conjunto de vértices, y P es el conjunto de aristas. Denotaremos la jerarquía de conceptos como J = (O, P).

Proporcionamos una estructura jerárquica a la organización del modelo conceptual, de forma que si existe un camino desde o_i a o_j diremos que o_j es descendiente de o_i, o que o_i es antecesor de o_j. A pesar de que las relaciones definidas mediante el conjunto P son las características de los árboles, J no es un árbol al no ser conexo[7]. Nosotros contemplamos la posibilidad de que un determinado concepto pueda tener varios antecesores, o que puedan existir nodos aislados, o incluso varias constelaciones independientes de conceptos (*bosques* en la terminología de la teoría de grafos y árboles). Cualquiera de estos motivos rompe con la definición de grafo conexo.

Semánticamente, un concepto descendiente de otro debe compartir los atributos de sus antecesores, a la vez que enriquece su definición con alguna o algunas características más. Tradicionalmente esta relación ha sido denominada *is-a* por la comunidad investigadora en Inteligencia Artificial, y tomando ese significado, nosotros tomamos como dirección de la arista la que va desde el nodo hijo al nodo padre.

Ejemplo 2.2 La figura 7.a muestra una posible organización jerárquica de un grupo de conceptos. El grafo de la figura 7.b no es válido por cuanto presenta un ciclo.

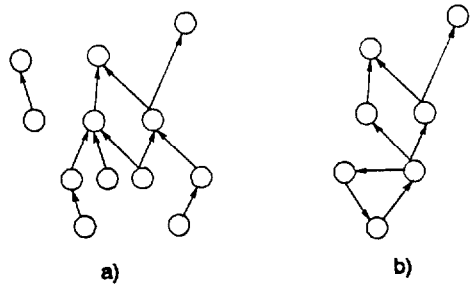


Figura 7

[7] Un grafo es conexo cuando existe un único camino entre cualquier pareja de vértices del grafo. Un grafo es un árbol cuando es conexo y no contiene ciclos [GRI89; ABE90].

Definición 2.5 Diremos que existe un camino dirigido entre dos nodos de la jerarquía o_i, o_j si existe una sucesión de aristas (o_i, o_k), (o_k, o_r), ...,$(o_l, o_j) \in P$ que parten de o_i y llegan a o_j. Especificamos el camino como pc_{ij}

Definición 2.6 El conjunto de antecesores del concepto o_i ($A(o_i)$) es el formado por todos los nodos para los que hay un camino dirigido que parte desde o_i y llega a dichos nodos.

$$A(o_i) = \{ o_j \in J \mid \exists\ pc_{ij} \}$$

Definición 2.7 La zona de influencia del concepto $o_i \in J$ la forman el conjunto de conceptos que pertenecen a los subárboles que tienen como nodos raíz a todos los antecesores directos de o_i, o, en el caso de que no existan antecesores de o_i, los nodos que pertenecen al subárbol que tiene como raíz al propio o_i, excluyendo siempre al nodo o_i.

$$Z(o_i) = \{ o_j \in J \mid \exists\ pc_{ji} \} \cup \{ o_r \in J \mid \exists\ p_{ir} \in P \} \cup \{ o_k \in J \mid \exists\ p_{ir} \in P \wedge \exists\ pc_{kr} \} - \{ o_i \}$$

Ejemplo 2.3 La figura 8 muestra cual sería la zona de influencia de un concepto que perteneciera a la situación actual

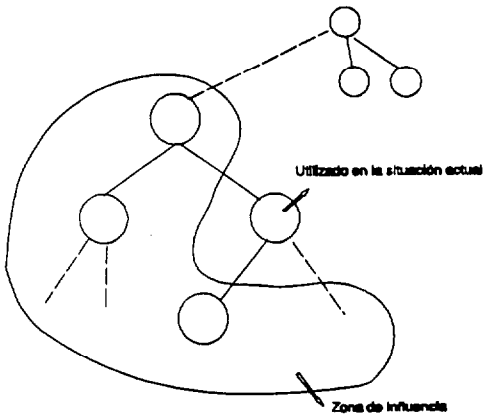


Figura 8

2.3 Modelo Causal

En esta sección se incluyen las definiciones relacionadas con la especificación del modelo causal del Dominio

Definición 2.8 Definimos dos tipos de nodos en el modelo causal: nodos OR y nodos AND. Un nodo OR del modelo causal se compone de dos expresiones relacionadas mediante alguno de los operadores relacionales clásicos (=, <>, >, >=, <, <=), y tiene la forma *Objeto.propiedad operador-relacional Objeto.propiedad.* Los operandos tienen la forma (o_i, a_j), donde $o_i \in O$, $a_j \in o_i$. Además, a_j puede hacer referencia a un atributo que calcula su valor mediante una fórmula. En cualquier caso, debe existir una compatibilidad en el tipo del resultado al que se evalúa la pareja objeto.propiedad (numérico o string).

Un nodo AND no tiene ningún componente excepto el identificador de tipo AND.

Definición 2.9 Si denotamos por N al conjunto de nodos causales, y $L \subseteq P \times P$. Entonces, una red causal *D* es el par (N, L), la cual es un grafo dirigido en N sin lazos, donde N es el conjunto de vértices, y L es el conjunto de aristas. Denotaremos la red causal como D = (N, L).

Dada una arista l_{ij}, que relaciona dos nodos n_i, n_j, decimos que el nodo n_j es consecuencia del nodo n_i, y que el nodo n_i es el antecedente y el n_j el consecuente de la relación l_{ij}. Para cada nodo n_i del grafo D, se define una lista de antecedentes formada por el conjunto de nodos de N que comparten relación con n_i de la forma (n_j, n_i), y de igual manera

Ejemplo 2.4 La figura 9 muestra un segmento de una red causal. En ella se pueden observar los dos tipos de nodos que pueden coexistir en la red. Los nodos OR han sido editados para facilitar la legibilidad.

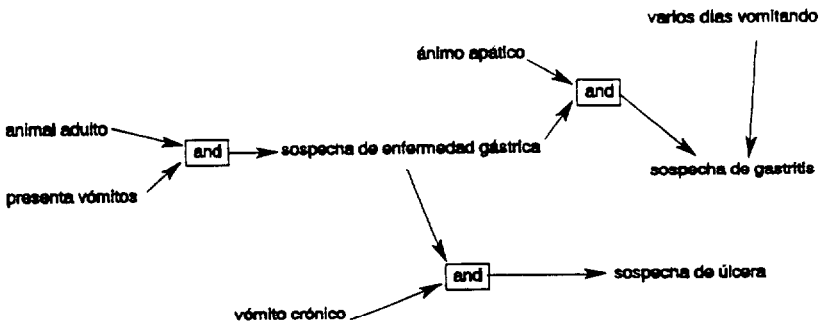


Figura 9

Definición 2.10 Dado el conjunto de nodos causales N, y el conjunto B formado por los elementos lógicos {verdadero, falso}. Definimos una relación f: N $\longrightarrow$ B, que por estar definida para todos los elementos de N, es una función. *f* la denominamos *estado de verificación del nodo.*

Si el nodo es de tipo OR, su valor resulta de sustituir cada pareja (o_i, a_j) que aparece en el nodo por los valores reales que toma para el caso que se esté considerando, y posteriormente evaluar la expresión relacional que contiene el nodo causal con los valores sustituidos. Entonces, para un $n_i \in N$ y de tipo OR

f(n) = verdadero sii se verifica la expresión que se define en n.
f(n) = falso cuando dicha expresión no se verifique.

Para determinar el estado de verificación de un nodo de tipo AND, será preciso evaluar el conjunto de antecedentes del nodo AND. Diremos que f será verdadero cuando todos los nodos antecedentes del nodo AND tienen un estado de verificación verdadero.

$f(n_i)$ = verdadero sii $\forall (n_j, n_i) \in L$, $f(n_j)$ = verdadero
$f(n_i)$ = falso en otro caso

Definición 2.11 Dado el conjunto de aristas L del grafo D = (N, L), y el conjunto B formado por los elementos lógicos {verdadero, falso}. Definimos una relación g: L $\longrightarrow$ B, que por estar definida para todos los elementos de N, es una función. A *g* la denominamos *estado de verificación de la relación causal.* Dado un $l_{ij} = (n_i, n_j) \in L$, diremos que $g(l_{ij})$ es verdadero cuando se verifica simultáneamente que $f(n_i)$ y $f(n_j)$ son también verdaderos. Al igual que la función *f*, *g* utiliza los valores almacenados en los casos para su evaluación. Siempre nos referiremos a *g* en relación con el caso que le proporciona dichos valores.

Definición 2.12 Los nodos AND presentan un comportamiento característico que puede ser modelado mediante una propiedad sobre dichos nodos. A diferencia de los nodos OR, un nodo AND debe tener una lista de antecedentes y de consecuentes no nula.

Sean na = $\{n_1, n_2, ..., n_k\}$ el conjunto de antecedentes y n_c un nodo perteneciente al conjunto de consecuentes del nodo AND n_a. Entonces, decimos que existe una relación causal entre el conjunto na y el nodo n_a, que lo representamos como $l_{1,2,...n;\,a} = (\{n_1, n_2, ..., n_k\}, c_a)$. A esta propiedad la denominamos transitividad de la relación AND.

Si ahora $g(n_i)$ = verdadero $\forall n_i \in$ na, y $g(n_a)$ = verdadero, entonces el estado de verificación de la relación causal $l_{1,2,...n;\, a}$ será verdadero. Y recíprocamente, será falso en cualquier otro caso.

Definición 2.13 Un nodo $o_i \in J$ pertenece a la red causal D si alguna pareja (o_i, a_j) aparece como operando en algún nodo de D.

2.4 Memoria de Casos

Una memoria de casos se compone de una colección de episodios o casos resueltos. Cada caso de memoria deberá hacer referencia a los objetos que utiliza, especificando valores concretos para los atributos que utiliza. Además, un caso mencionará a determinados conceptos que poseen una relevancia especial para su resolución. Estos objetos se asocian normalmente con hipótesis parciales que se van generando durante la resolución de un problema determinado. Se corresponden las alternativas que se consideraron, y por eso se suelen denominar conceptos 'estrella' o simplemente 'metas'. El conjunto de metas asociadas con un caso concreto permiten reconstruir su *traza derivacional*, o lo que es lo mismo, el camino que se siguió durante su resolución. Las siguientes definiciones describen los elementos que componen un caso de memoria.

Definición 2.14 Una tripla es la tupla (o_i, a_j, v_k) donde $o_i \in O$, $a_j \in o_i$, y v_k es uno de los valores que puede recibir el atributo a_j en el concepto o_i. T es el conjunto de todas las triplas que se pueden definir en el Sistema. Podemos definir múltiples conjuntos $T' \subset T$ tal que dadas dos triplas cualquiera del conjunto T' no coinciden simultáneamente en los dos primeros componentes de la tupla. Es decir:

$$\nexists\, (o_i, a_j, v_k), (o_i', a_j', v_k') \in T', \text{ tal que } o_i = o_i', a_j = a_j', v_k \neq v_k'$$

Al conjunto T' lo denominamos conjunto consistente en T

Definición 2.15 Dos triplas (o_i, a_j, v_k), (o_i', a_j', v_k') son equivalentes ($\equiv$) si $o_i = o_i'$, $a_j = a_j'$, y $h(v_k, v_k') =$ verdadero. Donde la función h evalúa la equivalencia sintáctica de los valores v_k y v_k'. La función *h* puede tomar los valores {verdadero, falso}, y debe tener en

cuenta el tipo del atributo para el que está evaluando su valor. No es lo mismo comparar dos valores numéricos que dos cadenas de texto. Las posibilidades de implementación de la función *h* para los valores numéricos pueden variar desde una rigidez total (los dos valores deben ser iguales), pasando por la definición de una ventana de flexibilidad que dependa del rango de a_j, hasta la asignación de una expresión matemática a *h* en la que se tengan en cuenta otra serie de factores semánticos (el concepto que se está considerando o cualquier otra característica).

Ejemplo 2.5 Las gráficas de la figura 10 muestran algunas posiblidades de medir la similaridad sintáctica cuando el tipo del atributo es numérico. En a) la función h sólo será verdadera si coinciden ambos números; para la opción b) se define una ventana (en función del rango de valores posible, por ejemplo) y un valor v_k' será equivalente a uno v_k si está a una distancia menor que la definida por la ventana. La opción más compleja se representa en la gráfica c). El valor de H lo calculará una función más compleja que dependa de v_k y v_k'.

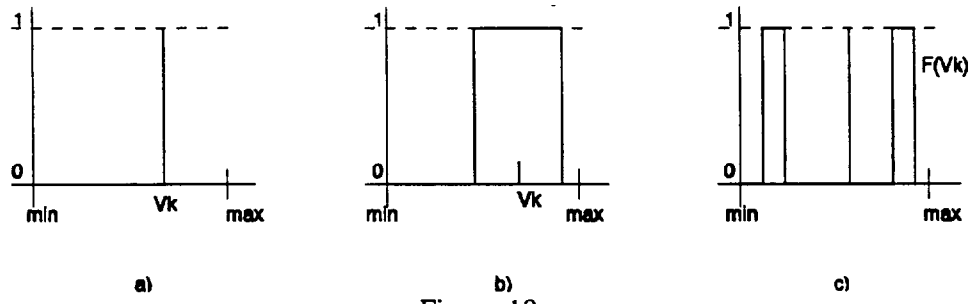


Figura 10

Definición 2.16 Las metas incluyen en su especificación tanto al concepto que permite su definición, como el motivo que justifica su verificación para cada caso que la utiliza y en la que se verifica. Es decir una meta *m* tendrá dos partes: el identificador, y su justificación causal. Lo cual podemos denotar por $m = (<o_i, a_j> , n_k)$ donde $o_i \in O$, a_j es el atributo que identifica al objeto, y $n_k \in D \cup \{null, \varnothing\}$. La justificación de una meta puede ser el conjunto vacío significando que dicha meta no encontró justificación causal, y que por lo tanto, se trata de una alternativa considerada y que no tuvo éxito. El valor 'null' indica que no es necesaria una justificación causal para verificar dicha meta. M es el conjunto de todas las metas que se pueden definir en el Sistema. Al igual que con las triplas, podemos definir conjuntos D' en D, de manera que no coexistan dos metas con el mismo identificador:

$$\nexists\ (<o_i, a_j> , n_k), (<o_i', a_j'> , n_k') \in D', \text{ tal que } o_i = o_i', a_j = a_j', n_k \neq n_k'$$

Definición 2.17 Un caso se compone obligatoriamente de una Unidad de Información de identificación (UII), de un conjunto arbitrario no nulo de triplas, y de un conjunto de metas con al menos un elemento. Podremos referirnos a un caso concreto de la base como $c_i = (UII_i, T_i, M_i)$, donde UII_i es una UI con características especiales, $T_i \subset T$, $M_i \subset M$, donde T_i y M_i son conjuntos consistentes en T y M respectivamente.

La UII es una unidad de información que contiene información de identificación del caso, y que tiene únicamente un carácter documental. Puede contener información como un título, categoría a la que pertenece, y una descripción en lenguaje natural de las características del caso. La definición explícita de esta estructura de información dependerá de cada aplicación y se deja a cargo del desarrollador. Debido a esta falta de estructuración, esta información no será utilizada durante el proceso de resolución.

Para cada meta ($<o_i, a_j>, n_k$) que pertenece al caso, existe una tripla asociada (o_i, a_j , v_r) que también pertenece al caso. Cuando el caso es la situación actual, nos referiremos a él como *Ca*, y cuando se trate de un caso de la base c_i (o *Ci*). Una tripla puede llegar a pertenecer a un caso por varios caminos:

- Especificando la tripla directamente en la definición del caso
- Indirectamente cuando se trata de una tripla que calcula su valor evaluando la expresión que se especifica en el campo 'fórmula' del atributo, el cual sólo debe hacer referencia a información que es conocida para el caso en cuestión.
- Durante la ejecución del Sistema, puede requerirse del usuario que proporcione determinada información para enriquecer Ca.
- Durante la ejecución del Sistema, cada vez que se verifica una meta, la tripla asociada esta se incluye en Ca con el valor que tenía en la situación de memoria de la que se toma la meta, o, si se trata de una fórmula, con el valor que resulte de la evaluación de esa expresión.

Cualquier tripla o combinación de ellas que pertenezcan a un caso de memoria pueden constituir la información de *solución* del caso. Cada uno de estos segmentos de la solución debe codificarse también en el conjunto de metas del caso (aunque sea con valor de justificación null). Debe existir en cada caso al menos una tripla que represente a la solución del caso. Por eso, tanto el conjunto de triplas como de metas no pueden ser nulos.

Ejemplo 2.6 El caso que se muestra a continuación pertenece al dominio de la concesión de créditos personales. En él se pueden apreciar claramente las tres partes de las que consta un caso: la identificación, el conjunto de triplas, y el de metas. Algunos atributos calcularán su valor mediante las

fórmulas que tendrán asociados, como pueden ser la cuota mensual o el margen de garantía. Otros tomaron su valor de los datos suministrados por el usuario, y otros se infirieron durante la resolución, como la necesidad de aval.

Identificación:
Id: c-14
Categoría: concedidos con aval

Triplas:
- Préstamo
  - cantidad: 1.750.000
  - plazo: 24 meses
  - propósito: estudios
  - estado: concedido
  - cuota: 85.820 ptas. *fórmula
- Trabajo
  - Contrato: true
  - Temporal: false
  - Salario: 225000 ptas.
  - Ocupación: Funcionario
  - Años trabajando: 10
- Cliente
  - edad: 45
  - Casado: true
  - Años casado: 20
  - Hijos: 1
  - Propiedades: true
  - casa propia: true
  - Gastos: 130000 ptas.
  - margen: cliente.gastos * 0.2
- Aval
  - estado: verificada
- Solvencia
  - estado: verificada
- Estabilidad personal
  - estado: verificada
- Carrera profesional
  - estado: verificada

Metas
- Aval, estado, nodocausal asociadoX
- Solvencia, estado, nodocausal asociadoY
- Carrera profesional, estado, nodocausal asociadoZ
- Estabilidad personal, estado,nodocausal asociadoV
- Crédito, estado, null
- Crédito, contribución mensual, null

Definición 2.18 Pertenencia a un Caso.

Una pareja (o_i, a_j) pertenece a un caso si este tiene una tripla que nombra al mismo concepto y al mismo atributo.

$$(o_i, a_j) \in Ci, \quad \text{sii} \quad o_i \in J, a_j \in o_i, \text{ y } \exists\, (o_i', a_j', v_k) \in Ci \quad \text{t.q. } o_i = o_i' \text{ y } a_j = a_j'$$

Un concepto o_i pertenece a un caso si existe una tripla perteneciente al caso que contiene dicho concepto.

$$o_i \in Ci, \quad \text{sii} \quad o_i \in J, \text{ y } \exists\, (o_i', a_j, v_k) \in Ci \quad \text{t.q. } o_i = o_i'$$

Un nodo causal pertenece a un caso, si este únicamente contiene elementos que son conocidos para el caso.

$$d_i \in Ci, \text{ sii } d_i \in D, \forall (o_j, a_k) \in d_i \ \exists (o_i', a_j', v_k) \in Ci \text{ t.q. } o_i \in \{o_i'\} \cup A(o_i') \text{ y } a_j = a_j'$$

Una relación de la red causal pertenece a un caso si el antecedente y el consecuente de la relación también pertenecen al caso.

$$l_{ij} \in Ci, \text{ sii } l_{ij} \in L, d_i, d_j \in Ci$$

Definición 2.19 La unión de dos casos (Ci ∪ Cj) da como resultado otro caso compuesto por las triplas de Ci más las de Cj que no coinciden simultáneamente en el objeto y atributo con alguna de las de Ci. Cuando dos triplas coinciden en el objeto y atributo, se tomará el valor de la tripla que pertenezca al primer operando de la operación unión.

$$Ci \cup Cj = \{ (o_r, a_p, v_k) \in Ci \wedge (o_r', a_p', v_k') \in Cj \} - \{ (o_r', a_p', v_k') \in Cj \mid \exists (o_r, a_p, v_k) \in Ci \wedge o_r = o_r', a_p = a_p' \}$$

De la expresión anterior es fácil deducir que esta operación no es conmutativa.

Definición 2.20 La intersección de dos casos (Ci ∩ Cj) produce un nuevo caso que estará compuesto por las triplas de Ci que provocan un incremento positivo al calcular FS(Cj), donde FS es la función que mide la similaridad entre la situación Cj y Ci, y que será definida en una próxima sección.

$$Ci \cap Cj = \{ (o_r, a_p, v_k) \in Ci \mid FS(Cj)_{(or, ap, vk)} > 0 \}$$

Al igual que la unión, la intersección no es una operación conmutativa.

Definición 2.21 Una memoria de casos es un conjunto de situaciones C_i. Denotamos a una base de casos por la letra E.

Definición 2.22 La estructura representacional del sistema que estamos especificando es la tupla formada por la memoria de casos, la jerarquía conceptual, y el modelo causal: (E, J, D).

3 Control

El objetivo del mecanismo de control del Sistema es el de alcanzar soluciones. Dada una situación de entrada Ca, enriquecer dicha descripción con la solución de la situación, y si fuera posible, indicar el camino que se debe seguir para resolver el problema. Este proceso lo hemos descompuesto en dos partes:

- Recuperación
- Adaptación

En la recuperación seleccionamos el conjunto de situaciones que en principio son candidatas a aportar su solución. Utilizaremos una función de evaluación de la similaridad durante esta fase. Posteriormente será necesario asimilar el camino de resolución que se siguió en cada una de las situaciones recuperadas. Un caso aportará su solución si el conjunto de metas que contiene son verificables con la información que representa a la situación actual. A este proceso lo denominamos adaptación. Durante la adaptación, la descripción de Ca se enriquecerá con nueva información, lo que puede modificar los índices de similaridad de los casos de la base. O dicho de otra forma, bajo determinadas circunstancias pueden ser necesarias múltiples iteraciones sobre las fases de recuperación y adaptación. La figura 11 refleja este proceso.

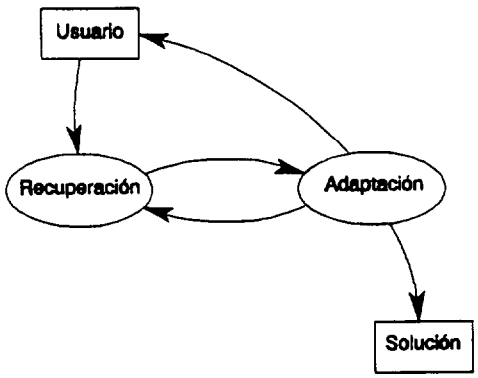


Figura 11

Esta sección la hemos dividido en 2 apartados relacionados respectivamente con la definición de la función de similaridad, y con el proceso de resolución de problemas.

3.1 Función de Similaridad (FS)

La Función de evaluación de la similaridad entre dos situaciones se define en base a 8 conjuntos que son los que definiremos a continuación. Siempre nos referiremos a la situación actual como Ca, y a un caso de la base como Ci. Los conjuntos que utiliza la FS siempre estarán asociados a los casos de la base.

Cada uno de estos conjuntos se utiliza para evaluar un aspecto concreto de la similaridad entre las dos situaciones. Podemos clasificar estos aspectos en tres categorías según debamos utilizar la jerarquía de conceptos, la red causal, o la propia memoria de casos. Se trata de descomponer cada UI presente en la situación actual en distintas dimensiones, una por naturaleza de la información que codifica.

Dentro de cada categoría distinguiremos dos tipos de conjuntos: aquellos que se encargan de identificar un aspecto de forma directa (similaridad directa), y los que en ausencia de una similaridad directa deben buscar fuentes alternativas de similaridad para evaluar la misma característica de la información (similaridad alternativa). Queremos resaltar que la similaridad alternativa sólo tiene sentido cuando no existe una similaridad directa entre dos casos en relación con la dimensión que están evaluando.

De esta forma definiremos 8 conjuntos: los dos primeros se referirán a la jerarquía de conceptos, los 4 siguientes a la red causal, y los dos últimos a la base de casos.

Definición 3.1 El conjunto D1 es el conjunto formado por todos los conceptos de la Jerarquía que pertenecen simultáneamente a Ca y a Ci. O sea,

$$D1 = \{ o_j \in J \mid o_j \in Ca \wedge o_j \in Ci \}$$

Ejemplo 3.1 En la figura 12 se puede apreciar como el conjunto D1 está compuesto por los elementos comunes a ambas situaciones.

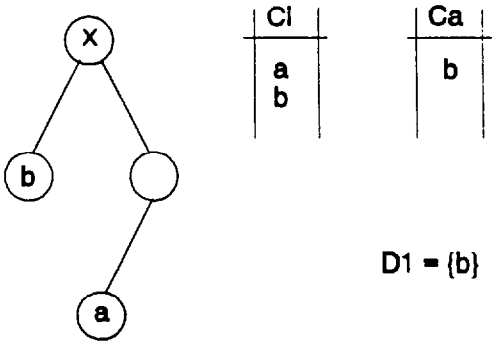


Figura 12

Definición 3.2 El conjunto D2 es el conjunto formado por todos los conceptos de la jerarquía y de Ca que no pertenecen al conjunto D1 y que tienen una zona de influencia con Ci no nula.

$D2 = \{ o_k \in Ca \mid o_k \notin D1 \wedge Q(o_k)_{Ci} \neq \{\varnothing\} \}$ donde Q es el conjunto de objetos de Ci que pertenecen a la zona de influencia de o_k.

Ejemplo 3.2 Ahora, para calcular el conjunto D2 es preciso calcular la zona de influencia del concepto 'a'. Como dicho conjunto no es nulo, 'a' formará parte de D2.

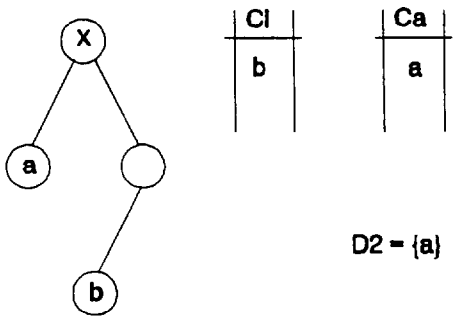


Figura 13

Definición 3.3 D3 lo forman los conceptos que pertenecen a Ca, o sus antecesores, que aparecen nombrados en algún nodo causal, y que también pertenecen a Ci o son alguno de sus antecesores.

$$D3 = \{ o_i \in Ca \mid o_i \in Ci, \text{ y además } o_i \in D\} \cup$$
$$\{ o_i \in Ca \mid \exists o_j \text{ antecesor de } o_i \text{ y de cualquier } o_k \in Ci, \text{ y además } o_j \in D\}$$

Ejemplo 3.3 En el ejemplo de la figura 14, D3 es la intersección de dos conjuntos: el de los antecesores de 'b' y de 'a'.

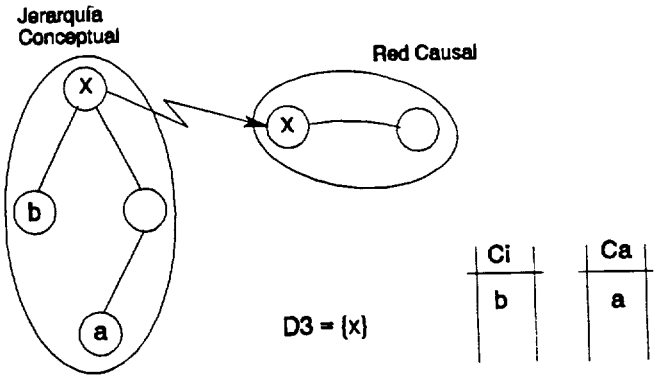


Figura 14

Definición 3.4 D4 son las relaciones de Ca que también se verifican en Ci.

$$D4 = \{ l_{ij} \in Ca \mid l_{ij} \in Ci, \ \wedge g(l_{ij}) = \text{verdadero tanto para Ca como para Ci}\}$$

Ejemplo 3.4 La figura 15 muestra una relación causal que se verifica simultáneamente para las situaciones Ca y Ci. Ese es precisamente el conjunto D4.

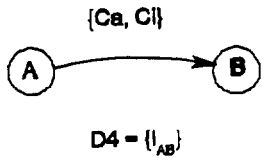


Figura 15

Definición 3.5 D5 está compuesto por las relaciones que se verifican en Ca, cuyo consecuente forma parte de alguna otra relación que se verifica en Ci y que no pertenecen al conjunto D4.

$D5 = \{ l_{rj} \in Ca \mid \exists\, l_{kj} \in Ci, \wedge\, g(l_{rj}) = \text{verdadero para Ca} \wedge g(l_{kj}) = \text{verdadero para Ci} \wedge r \neq k \wedge l_{rj} \notin D4\}$

Ejemplo 3.5 Si tenemos una relación l_{ab} que se verifica en Ca y otra l_{cb} en Ci, ambas están compartiendo las mismas consecuencias causales.

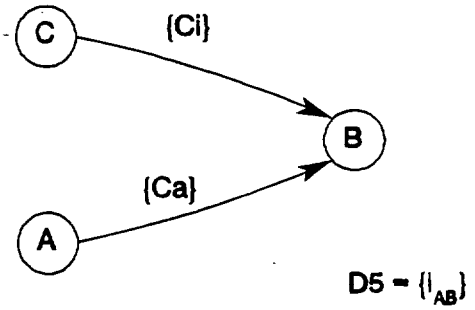


Figura 16

Definición 3.6 D6 es el conjunto formado por las relaciones de Ca que se verifican, cuyo consecuente es, a su vez, antecedente de una relación que se verifica para Ci y que no pertenecen al conjunto D4.

$D6=\{l_{rj} \in Ca \mid \exists l_{jk} \in Ci, \wedge\, g(l_{rj}) = \text{verdadero para Ca} \wedge g(l_{jk}) = \text{verdadero para Ci} \wedge l_{rj} \notin D4\}$

Ejemplo 3.6 En este ejemplo, se muestra como un nodo causal puede ser un nexo entre dos relaciones, indicando, tal vez, un posible camino a explorar.

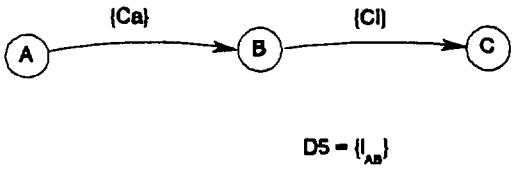


Figura 17

Definición 3.7 D7 lo forman las triplas que tienen en común Ca y Ci.

$$D7 = \{ (o_r, a_j, v_p) \in Ca \mid \exists(o_r', a_j', v_p') \in Ci \wedge (o_r, a_j, v_p) \equiv (o_r', a_j', v_p')\}$$

Ejemplo 3.7 Si Ca está compuesto por las triplas:
(animal, edad, adulto)
(animal, raza, bulldog)
y Ci por:
(animal, edad, adulto)
(animal, color, gris)
entonces el conjunto D7 está formado por la tripla (animal, edad, adulto)

Definición 3.8 D8 lo forman las parejas (o_r, a_j) que tienen en común Ca y Ci que no pertenecen a D7

$$D8 = \{ (o_r, a_j) \in Ca \mid (o_r, a_j) \in Ci \wedge (o_r, a_j) \notin D7\}$$

Ejemplo 3.8 Si Ca está compuesto por las triplas:
(animal, edad, adulto)
(animal, raza, bulldog)
y Ci por:
(animal, edad, cachorro)
(animal, color, gris)
entonces el conjunto D8 está formado por la pareja (animal, edad)

De los conjuntos así definidos, D1, D4 y D7 son los que se corresponden con la valoración de la similaridad directa; y D2, D5, D6 y D8 son los relacionados con la similaridad alternativa. El conjunto D3 facilitará que aquellos conceptos que verifiquen su definición sean considerados como conceptos de especial relevancia.

Definición 3.9 Sea CD el conjunto formado por {D1, D2, D3, D4, D5, D6, D7, D8}. La función de similaridad FS se define como una aplicación entre cada uno de los casos de la base y Ca de la forma

$$FS : E \rightarrow Z^{+} \cup \{\varnothing\}$$
$$c_i \rightarrow \sum_{j=1}^{8} W_j \ |D_j|$$

donde $c_i \in E$, $D_j \in CD$, y los $W_j \in Z^{+}$ son pesos de ponderación de cada una de las dimensiones de la fórmula. FS debe calcularse para todos los casos de la base cada vez que se plantea un nuevo problema en el sistema. Así, si declaramos P_j como el cardinal de D_j podemos representar el valor de la función de similaridad en cada caso de memoria, teniendo como entrada al sistema a la situación Ca como:

$$FS(c_i)_{Ca} = \sum_{j=1}^{8} W_j \ P_{ij}$$

Los W_j son pesos que nos permiten ajustar el funcionamiento de la función de similaridad, y sus valores dependen del conocimiento que se tenga del dominio en cada momento. O dicho de otra forma, los w_j deben especificarse para cada aplicación que se desarrolle con el Sistema. En el capítulo III se desarrollan diversos aspectos relacionados con estos pesos. Evidentemente, los pesos asociados con la similaridad alternativa deben ser menores que los asociados con la directa. En caso contrario, podría darse la circunstancia de que un caso de la base que coincida al 100% con Ca obtenga un valor de FS inferior a otro cualquiera de la Base, lo cual claramente es un error.

3.2 Proceso de Resolución

Podemos dividir el proceso de resolución en dos componentes principales: el primero construiría un conjunto de situaciones que, en principio pueden aportar su solución. El segundo trataría de incluir las metas que se verifican para cada uno de los casos recuperados en la situación actual. Comenzaremos definiendo cada uno de estos problemas por separado, y seguiremos con una elaboración completa del proceso de resolución.

Definición 3.10 El conjunto de recuperación es el formado por todos los casos de la base cuya similaridad con la situación actual les permite diferenciarse del resto de los casos definidos en la memoria de situaciones. De una manera formal podemos escribir:

$CR = \{ c_i \in E \mid FS(c_i)_{Ca \cup X} > FS(c_j)_{Ca \cap Ci \cup X} \forall i \neq j, \forall X \in \wp(c_j - c_i) \cup \{\varnothing\} \wedge FS(c_i)_{Ca} > UMS \}$
donde $\wp$ simboliza el conjunto de 'las partes de...'

El primer término nos asegura que Ci no va a ser superado con la información que comparte con los otros casos de la base, y es el mejor representante con esa información. El segundo nos permite comprobar que la parte de la descripción de Ca que tiene Ci es significativa.

Definición 3.11 Un caso de la base aporta su solución a la situación actual si pertenece al conjunto de recuperación y todas las metas que contiene se verifican para el caso actual.

$c_i \in E$ aporta su solución, sii $c_i \in CR \wedge \forall(<o_i, a_j> , n_k) \in c_i$ con $n_k \neq \{\varnothing\}$, la meta $<o_i, a_j>$ se puede verificar para Ca.

Algoritmo de Resolución

Las anteriores definiciones las podemos integrar en un único proceso de funcionamiento del Sistema. El comportamiento de este proceso puede variar sensiblemente en función de que deseemos encontrar todas, o sólo una (la mejor) solución. En principio, supongamos que queremos encontrar todas las soluciones posibles con los datos de entrada.

0. Inicializar Ca = {∅}, leer datos del usuario

1. Si existen datos del usuario, Incluir en Ca los datos del usuario y elaboraciones que puedan llevarse a cabo con esa información
   Sino, Finalizar.

2. $\forall c_i \in E$ calcular $FS(c_i)$ y construir CR

3. Si CR = {∅} leer más datos del usuario, ir a 1

4. Sino
   - 4.1 $\forall c_i \in CR$ comprobar si c_i aporta su solución a Ca
   - 4.2 Si $\nexists$ c_i que aporte su solución, leer datos del usuario, ir a 1

Cuando sólo se persigue una única solución, el paso 4.1 debe sustituirse por:

4.1 Mientras no exista una solución y CR ≠ {∅} hacer
seleccionar el c_i de mayor FS
comprobar si c_i aporta solución a Ca
Fin mientras

Comprobar que una determinada meta se verifica en la situación actual no exige necesariamente que se tenga que utilizar la misma justificación causal que para el caso con el que se está trabajando. Existen otras alternativas que son las que se exponen al descomponer dicha tarea:

1. $\forall(<o_r, a_j> , n_k) \in c_i$
   - 1.1 Comprobar si $(<o_r, a_j> , n_k)$ es verificable en Ca: $(<o_r, a_j> , \emptyset) \notin$ Ca
   - 1.2 Si es verificable, realizar sólo lo primero que se cumpla
     - a. Si $(<o_r, a_j> , n_k)$ se verifica en Ca, incluir la meta en Ca
     - b. Buscar el c_p t.q. FS(c_p) sea máximo, y $\exists(<o_r, a_j>, n_k') \in c_p$ que se verifique en Ca. Incluir esa meta en Ca
     - c. Buscar un $n_k'' \in$ D t.q. $(<o_r, a_j> , n_k'')$ se verifique en Ca. Incluir esa meta en Ca.
   - 1.3 Si no es verificable o no se cumple alguna de las opciones anteriores, incluir $(<o_r, a_j> , \emptyset)$ en Ca, y eliminar c_i de CR.

La ordenación que se muestra para las alternativas del paso 1.2 permiten una convergencia más rápida hacia la solución al premiar a aquellas situaciones que tienen más posibilidades de aportar su solución.

Abreviaturas utilizadas en este capítulo

Unidad de Información

UI	Unidad de información
UII	Unidad de información de identificación
o	Concepto
a	Atributo
v	Valor de atributo

Jerarquía de conceptos

J	Jerarquía de conceptos
P	Conjunto de aristas de la Jerarquía
A	Conjunto de antecesores
Z	Zona de influencia del concepto

Red Causal

D	Red Causal
N	Nodos causales
L	Aristas causales
f	estado de verificación del nodo causal
g	estado de verificación de la relación causal

Memoria de Casos

E	Memoria de casos
c	caso

Control

FS	Función de Similaridad
D1..D8	Conjuntos de la función de similaridad
Cr	Conjunto de Recuperación

Capítulo III

Diseño y Trabajo Experimental

En este capítulo se discutirán las principales decisiones de diseño que se tomaron durante la realización del trabajo experimental, y que dieron lugar a la descripción del Sistema del capítulo anterior. Las hemos agrupado en distintas secciones las cuales se apoyan siempre en las estructuras del capítulo anterior. Comenzaremos por las decisiones que implican directamente a las estructuras de representación y de inferencia; continuaremos con la función de similaridad; con diversos aspectos acerca del funcionamiento del Sistema; con el ajuste de la Función de Similaridad; y acabaremos comentando algunos aspectos acerca del rendimiento global del sistema frente a distintas situaciones.

Contenidos

1 Representación y Control

La arquitectura que se especificó en el tema 2 es fruto de una serie de decisiones que serán las que se expongan aquí. Muchas de esas decisiones tienen origen en el estudio realizado sobre algunos de los sistemas que actualmente existen y que se expusieron en el capítulo 1. Al final de dicho capítulo se comentaron algunas de las características que este sistema debe poseer, así que no redundaremos nuevamente en su justificación. Aquí nos limitaremos a mostrar como articulamos los distintos aspectos del Sistema que allí se introdujeron.

La propiedad fundamental de un sistema CBR es el razonamiento analógico. Este debe ser capaz de inferir similaridades a partir de las características que proporciona el usuario. Sin embargo, en el capítulo 1 quedó claro que estas características son insuficientes para construir un proceso analógico completo. Este debe trabajar a un nivel de abstracción adecuado. No es eficiente utilizar analogías con características de bajo nivel solamente, ni es conveniente trabajar con analogías demasiados abstractas que, debido a su generalidad, pierden su capacidad de discriminación. En cualquier caso, lo que parece necesario es, primero un mecanismo de abstracción de la información: este lo conseguimos mediante la organización jerárquica del modelo conceptual del dominio. Y en segundo lugar, una estructura que fije el nivel de abstracción adecuado a cada situación. Evidentemente ni todas las abstracciones son importantes, ni siempre una determinada abstracción es igualmente determinante. Este nivel lo fija el modelo causal del dominio que está enlazado con el conceptual según se ilustra en la figura 18. Estos enlaces se pueden situar a cualquier nivel dentro de la escala de abstracción. La información elaborada común a dos situaciones podrá ser, como se muestra en la figura, una relación causal que en principio no nombra a alguno o a ninguno de los conceptos de bajo nivel utilizados para describir a los dos casos.

El razonamiento analógico viene también enriquecido por la utilización de información elaborada y que sigue siendo de un nivel de abstracción al menos tan bajo como el dato que describe una nueva situación. Nos referimos a la utilización de la *cercanía* entre dos conceptos. Consideramos que la definición de *cercanía* proporcionada en el tema anterior mejora la capacidad analógica del Sistema al proporcionar un mecanismo alternativo de coincidencia entre dos situaciones. En cualquier caso, este es un aspecto secundario que puede ayudar en la valoración de la similaridad, pero que nunca podrá ser determinante debido a su menor carga semántica.

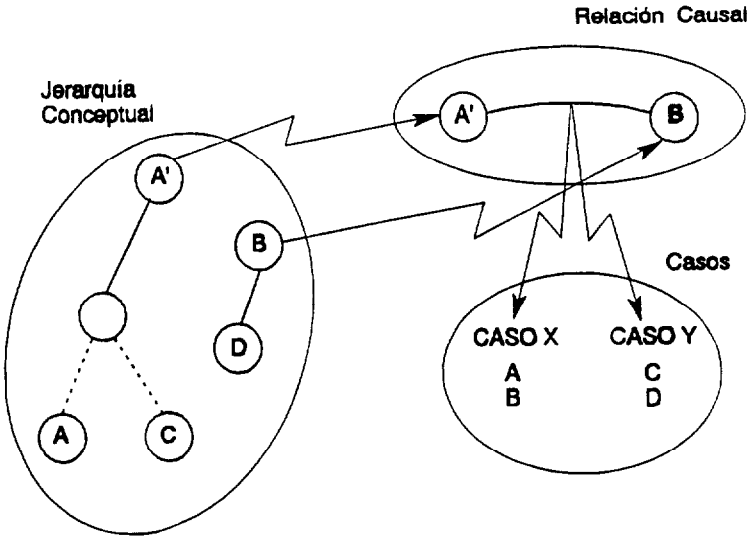


Figura 18. Similaridad causal mediante la abstracción de la información.

Otro aspecto importante del razonamiento analógico es su capacidad de proporcionar la importancia adecuada a *cada factor* en *cada situación*. Consideramos que hemos conseguido evitar la asignación subjetiva de 'pesos' a cada característica definida en el sistema, ya sea un peso por característica o uno por característica dentro de cada caso. La relevancia de un concepto para un caso determinado (un contexto) vendrá determinada por el número de enlaces que sea capaz de verificar por alguno de los siguientes medios:

- por la conexión con otros conceptos de mayor nivel de abstracción.
- por la conexión con nodos causales, ya sea de forma directa, o indirecta por medio de los conceptos de mayor nivel a los que está asociado.
- por la conexión con conceptos *cercanos* según la definición dada en el capítulo 2.
- por la utilización de definiciones de conceptos similares a las usadas en otras situaciones

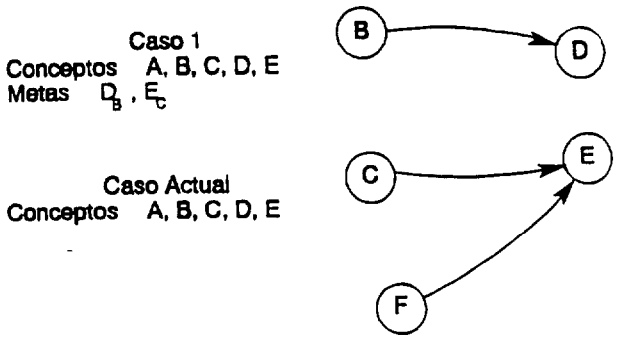
Teniendo en cuenta este conjunto de factores, se puede afirmar que en la valoración de cada concepto intervienen todas las estructuras de conocimiento del Sistema, dando lugar a una valoración que depende no sólo de la situación actual sino también de la posición que ocupa el concepto en la teoría global del dominio.

En relación también a la importancia relativa de cada concepto, también es necesario considerar la influencia que tiene un concepto definido para una situación sobre otra con la que comparte parte de la misma definición. Esta influencia será más importante cuanto más lo sean los atributos que tienen en común. Y viceversa: la pérdida del valor que aporta un concepto definido para una situación en otra situación, será mayor cuanto mayor sea la importancia que tengan los atributos que no comparten ambas definiciones del concepto. De esta forma se consigue otro efecto deseado: que la valoración de la similaridad tenga en cuenta distinciones contextuales como por ejemplo el valor concreto que recibe un determinado atributo. Además, la valoración de la similaridad entre dos definiciones de un concepto toma una de ellas como referencia, en lugar de una definición neutra proporcionada por una estructura global al sistema, lo cual es un efecto positivo desde el momento que la estructura de referencia la proporciona la situación actual, y por consiguiente, el comportamiento del sistema será distinto en función de la información de entrada.

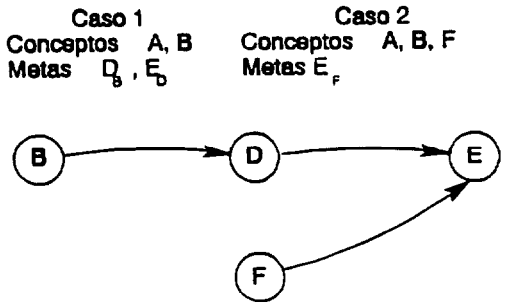
Otras dos cuestiones fundamentales son la reutilización de caminos de resolución, y la de segmentos de casos. Ambas son la justificación de la existencia de las *metas*. Las metas son hipótesis parciales que se corresponden con puntos intermedios de decisión, y que normalmente deben estar justificadas mediante algún enlace de tipo causal. Que un caso pueda ajustar su resolución a la que siguió un caso almacenado refuerza la decisión de recuperación de dicho caso. Las trazas derivacionales se almacenan usualmente como conocimiento causal local al caso. Nosotros proponemos unas estructuras globales de forma que se consiga una mayor reutilización del conocimiento codificado, y se facilite la exploración de caminos alternativos cuando la traza propuesta por el caso recuperado no pueda seguirse a partir de un determinado punto. Además, mejoramos la propuesta de analogía derivacional de Carbonell al permitir continuar una determinada traza, una vez que se ha salvado un obstáculo como la no verificación de una meta, utilizando la misma justificación que se usó en el caso alternativo. Por ejemplo, si para un caso actual como el de la figura 19a, se recupera el caso1, la construcción de la nueva vía de solución puede proceder infiriendo la meta D con la misma justificación que proporciona el caso1, aunque si el concepto C no tiene sentido en la nueva situación, tendrá que buscar una alternativa distinta a la que proporciona el caso1 para la meta E (como por ejemplo F). Si se consigue verificar esta meta, las conclusiones a las que se llegó con el caso1 seguirán siendo válidas para la situación actual.

Sin embargo, esta situación se complica cuando la justificación de una meta es otra meta. En la figura 19b se muestran dos casos de la base y la situación actual, además de un segmento de la red causal. Si para la situación actual se recupera como primera opción al caso2, y no se pudiera justificar la meta E en base a F, habría que saltar al caso1. Pero, como la meta E se justifica en este caso en base a una meta precedente D, habría que dotar al sistema causal de cierta capacidad de inferencia hacia atrás, al menos, a la hora de verificar las metas. Recordar que las metas son información que se deduce, y que usualmente no es

suministrada por el usuario. Así es como resolvemos la insuficiencia de la propuesta de Carbonell. Si no proporcionamos esta capacidad de inferencia backward, tendríamos que intentar adaptar el caso1 desde el principio hasta llegar a la meta actual. Al proporcionar una estructura causal global, ya es posible solventar este problema.



a)



b)

Figura 19. Reutilización del camino de resolución.

A la hora de seleccionar entre varios antecedentes candidatos a una meta, contemplamos las siguientes posibilidades:

a. seleccionar uno al azar

b. seleccionar aquel que suponga un menor coste para el usuario (e.d. que utilice información ya incluida en la situación actual, frente a aquel que requiera de nueva información)

c. seleccionar el que proponga el caso de memoria más cercano al caso actual que incluya la meta en cuestión (excluyendo al caso guía actual)

Nosotros nos decidimos por la última opción ya que esa política puede implicar la promoción de una caso que no supera los criterios de selección, a que sea solución. Por contra, cualquiera de las otras opciones puede premiar a casos que son irrelevantes para la situación actual, a la vez que impide que soluciones potencialmente factibles lleguen a serlo.

Además de las ventajas ya señaladas, la decisión de separar lo más claramente posible cada estructura de conocimiento se justifica también en un esfuerzo de facilitar las tareas del ingeniero del conocimiento y del experto humano a la hora de desarrollar aplicaciones CBR. Organizar el conocimiento utilizando estructuras simples y bien definidas evita ambigüedades a la vez que facilita la verificación y validación de la base de conocimiento. Reconocemos que tener que especificar las hipótesis parciales (o conceptos estrella) asociados con cada caso va en contra de nuestra política de simplicidad, pero no hemos encontrado otra alternativa si queremos guiar el proceso de resolución de problemas sin que se produzca una explosión combinatoria, y sin dar demasiada autonomía al sistema causal.

Finalmente, creemos que la coordinación entre las estructuras de conocimiento que proponemos facilita el desarrollo de sistemas incrementalmente, de manera que no exige una descripción exhaustiva de ninguno de los tres modelos para el que sistema funcione. Es más, siendo realista eso nunca podrá suceder, y creemos que los mecanismos que posee el sistema le permite trabajar eficientemente mediante alternativas a los posibles vacíos de conocimiento en alguna de las estructuras del sistema. Esa es la tarea de la función de similaridad, que comentaremos a continuación.

2 Función de Similaridad

La Función de Similaridad (FS) es el mecanismo de funcionamiento del sistema CBR. Para que el Sistema alcance una solución, la FS debe 'cortar' el campo de descripción de la Base de Casos tomando al caso actual como referencia. Para llevar a cabo este corte realizamos primero una transformación en el sistema de representación. Transformamos un sistema de referencia basado en el concepto como unidad de información hacia otro basado en el enlace. Esta transformación tiene las ventajas siguientes:

- Simplificar la recuperación al no tener que ponderar cada unas de las características definidas en el Sistema. La aportación de cada característica estará distribuida por las distintas dimensiones en las que el conocimiento se distribuye.
- Discriminar las distintas aportaciones semánticas de cada característica: conceptual, causal, y contextual.
- Facilitar el razonamiento analógico gracias a la utilización de dimensiones alternativas a aspectos semánticos concretos. De esta forma no se busca una característica alternativa, sino una unidad de información con las mismas características semánticas.

Las ocho dimensiones de la FS definidas en el capítulo anterior se pueden clasificar atendiendo al contenido semántico que codifican en: conceptuales, causales, y contextuales. Las primeras intentan reflejar coincidencias a nivel conceptual entre el caso actual y los casos de la Base. Las causales buscan distintos tipos de relaciones en común entre dos situaciones; y las contextuales buscan similaridades a nivel de descripciones de conceptos entre dos situaciones.

Las dimensiones denominadas D1, y D2 son las pertenecientes al primer grupo. Al igual que sucede con los otros grupos, una de las dimensiones registra similaridad directa, y la otra alternativa. D1 no deja de ser una heurística aunque es de sentido común. Se trata de responder a la cuestión: ¿qué se puede medir en una jerarquía cuando se dispone de una estructura de casos formada (esencialmente) de conceptos?. Creemos que su razón de ser es evidente. D2 busca alternativas cuando no se ha encontrado un enlace con las características de D1. En este punto se utiliza el significado de concepto *cercano* proporcionado en el capítulo 2. Es de esperar que los conceptos conceptualmente similares compartan cierta zona de influencia en la jerarquía. Si suponemos que el enlace que une dos conceptos es el típico *es-un*, dos hermanos en la jerarquía pertenecen al mismo conjunto, y ese es el significado de D2: la pertenencia a una misma categoría conceptual.

D3, D4, D5, y D6 son las dimensiones del grupo causal. Su cometido es el de evaluar el grado de relevancia causal de un determinado conjunto de características. Estas son las características del caso actual que aparecen en el modelo causal del dominio de la aplicación. La primera heurística de este grupo pretende premiar a aquellos conceptos que intervienen en el modelo causal. Existen unos conceptos que son más importantes que otros, y una forma de hacer esa distinción más evidente consiste en premiar a los que aparecen en alguna relación causal frente a los que sólo aparecen en la jerarquía y en el mejor de los casos, en alguna de las situaciones de memoria. D4 es, sin duda, la heurística de más peso en el sistema, ya que es la que proporciona una mayor carga semántica. Con D4 se buscan relaciones causales comunes entre la situación actual y los casos de memoria. Idealmente dos situaciones similares compartirán gran parte de su cadena causal, pero esto no tiene por qué suceder siempre, y por eso se hacen necesarios mecanismos alternativos de evaluación de una similaridad causal. El más evidente consiste en buscar caminos alternativos a una relación cuando esta no se verifica por cualquier motivo. D5 proporciona una vía alternativa para alcanzar un mismo consecuente, lo cual es casi equivalente a una relación directa. Y con D6 se exploran caminos que prolonguen la línea causal de razonamiento del caso actual. Con esta heurística se pretende guiar de forma inteligente la solicitud de información al usuario en el caso de que la descripción del caso actual estuviera incompleta. Solicitar información que puede facilitar la expansión hacia delante de la cadena causal es una buena alternativa. Con D6 se premia por anticipado a aquellas situaciones que verifican dichas relaciones.

El último grupo de dimensiones lo constituyen las dos últimas heurísticas. Decimos que son contextuales porque profundizan en la descripción de los conceptos que dos situaciones tienen en común. Dos conceptos pertenecen al mismo ámbito cuando están definidos utilizando los mismos descriptores (atributos y valores). D7 premia casos con este tipo de coincidencia (concepto, atributo, y valor). Como alternativa existe D8, que no tiene en cuenta el valor del atributo. Cuando lo único que coincide es el nombre del concepto, no es necesario incluir ninguna heurística más, al existir D1 que cubre esa posibilidad.

En definitiva, la valoración de la similaridad entre dos situaciones se descompone en una serie de aportaciones semánticas. Cada una de estas aportaciones se corresponde con alguna de las tres naturalezas del conocimiento que el sistema es capaz de distinguir: conceptual, causal, y contextual. Además se han articulado mecanismos alternativos de valoración de la similaridad que se ponen en marcha cuando no existe una similaridad 'directa'.

Realmente, estas dimensiones son el resultado de pruebas empíricas. Como tal, la función se podría clasificar dentro de los denominados *métodos débiles*. Sin embargo, consideramos que estas dimensiones trabajan eficientemente en común permitiendo extraer

la información más relevante que contiene el Sistema CBR en cada situación.

Una última cuestión a comentar acerca de la FS es el caso que se toma como referencia cuando se evalúa la FS. Existen dos opciones en esta interpretación:

1. La FS mide en qué medida está incluido cada caso de memoria en la descripción actual (Ca).
2. En qué medida está incluido Ca en cada caso de memoria

Para ilustrar estas dos opciones supongamos que tenemos los casos de memoria V, X, Y, y Z. El caso actual es Ca, y el vocabulario de representación es a, b, c, d, y e. Si cada caso viene representado tal como se muestra en la tabla 1, observamos que los porcentajes que se obtienen para cada opción son significativamente distintos (tabla 2)

Ca	V	X	Y	Z
a		a		a
b	b		b	b
c	c		c	c
			d	d
		e		

Tabla 1

	Ca	V	X	Y	Z
Valor Máximo[1]	30	20	20	30	40
Valor de coincidencia (match)		20	10	20	30
% match opción 1		66	33	66	100
% match opción 2		100	50	66	75

Tabla 2

Por ejemplo, con la opción 1, el caso X tiene sólo un descriptor en común (a) de los tres que describen al Ca. Por lo tanto se correspondería con un 33% como valor de la función FS. En cambio, para la opción dos necesitamos saber cual sería el valor máximo que podría alcanzar cada caso de memoria[2]. La fila denominada *valor máximo* de la tabla refleja este

[1] Por simplicidad en la exposición, suponemos que cada descriptor aporta una puntuación de 10 a la FS

[2] El valor máximo se obtiene cuando se consulta con un caso que es idéntico al de memoria.

valor. Sumando las aportaciones de Ca en cada uno de los casos obtenemos el valor de coincidencia, que en este caso es muy sencillo de calcular. La opción 2 se calcula como el porcentaje que representa el valor de coincidencia respecto al valor de coincidencia para cada caso.

¿Qué opción es mejor? Como puede observarse la opción dos posee un coste computacional bastante superior a la opción 1, ya que es necesario calcular el valor máximo para cada caso de memoria. Sin embargo esto no es demasiado problemático desde el momento en que se puede calcular en el momento de incluir el caso en la base, y no en cada consulta al Sistema. Además, la opción 1 puede dar lugar a ambigüedades en la selección del mejor caso, ya que por un lado impide evaluar la relevancia de la información que se proporciona con Ca, y a la vez permite que no se tenga en cuenta información del caso de memoria que podría ser vital. Esto se puede ilustrar con el siguiente ejemplo: si ahora tenemos un nuevo caso de memoria W={a,b,c,e}, con la opción 1 y con el Ca de la tabla 1 conseguiría un 100% de coincidencia, cuando realmente habría un conflicto con el caso Z que también obtiene el 100% de coincidencia. Si tomamos la opción dos ambos casos obtendrían un 75%, pero al menos ese valor nos indica que existe un margen del 25% por estudiar.

Otro efecto no deseado que tiene la opción 1 se produce cuando la descripción de la situación Ca se mejora durante la consulta. El comportamiento de los casos de memoria se puede ver reflejado con subidas o con bajadas del porcentaje de coincidencia. Si incluimos *e* en Ca, el porcentaje de X aumenta al 50% mientras que baja el del resto de los casos. Con la opción 2, sin embargo, cualquier adición a la descripción de Ca sólo modificará al alza los valores de coincidencias de los casos de memoria en el mejor de los casos. Esto facilita significativamente las tareas de evaluación de la solución tal como veremos posteriormente. También se consiguen mayores distinciones en la proporción que alcanza cada caso de memoria, al construirse el porcentaje en base a los valores máximos de cada caso, que se supone que son distintos. Estas distinciones son las que nos permiten 'afinar' el grado de similaridad entre Ca y los casos de la base (Ci).

3 Descripción de los mecanismos de funcionamiento

Las decisiones relacionadas con la operación del Sistema han sido las más complejas y las que han sufrido una mayor evolución desde la propuesta original. Nosotros consideramos que desarrollar algunos de estos aspectos cronológicamente permitirá obtener una mejor comprensión de las decisiones que finalmente se adoptaron.

El funcionamiento del sistema se puede organizar en varias fases: en la primera, se evalúa la FS con la información de la situación actual respecto a todos los casos de la base. Posteriormente, habrá que decidir si existe uno o más episodios que puedan ser considerados para su recuperación. Y finalmente se tendrá que comprobar la aplicabilidad de los casos recuperados a la situación actual mediante la asimilación del camino de resolución que se siguió en dichas ocasiones.

Nuestro estudio se ha apoyado principalmente en la determinación de las condiciones necesarias para que un caso pueda ser recuperado. Cuándo podemos decir que una determinada situación es potencialmente la solución, o una de las soluciones del Sistema.

3.1 Recuperación

El objetivo del proceso de recuperación es seleccionar un caso, el mejor caso. Para ordenar la base de casos respecto a Ca tenemos la función de similaridad FS. Sin embargo esto no es suficiente. El proceso de recuperación no siempre tendrá éxito. Cuando la información que describe a Ca sea irrelevante (como podría ser el nombre), o ambigua, resultando en más de un caso con puntuaciones muy similares, entonces no podemos garantizar que el proceso de recuperación ha seleccionado el caso adecuado.

Tradicionalmente, en sistemas de scoring como el que estamos proponiendo se ha venido utilizando el concepto de umbral para delimitar el punto a partir del cual un caso pasa a ser solución. Este umbral es un parámetro global al sistema y definido por el usuario. Por ejemplo, situar un umbral al 80% significa que el sistema habrá alcanzado una solución cuando alguno de los casos de memoria haya superado ese nivel de similaridad respecto a Ca. En sistemas con conocimiento causal, una solución potencial se valida comprobando que Ca

y el Ci correspondiente comparten las mismas características causales (almacenadas en Ci), o ejecutando una serie de pruebas sobre Ca que son características de Ci. En el tema 1 ya se han comentado estas aproximaciones, así que no profundizaremos más en ellas. Por los motivos que allí apuntamos, estas posibilidades se nos antojan demasiado ingenuas y poco realistas. Especialmente la de proponer un umbral global. Nuestra experiencia con la herramienta CBR Express que utiliza esta política no es satisfactoria. Incluso para pequeños prototipos, ajustar dicho nivel manualmente de manera eficiente es poco menos que imposible. Normalmente es necesario fijar umbrales muy altos para reducir el riesgo de error. Esto conduce a que habrán situaciones en que no se alcancen soluciones con unos descriptores que se podrían considerar como aceptables, mientras que en otras ocasiones se recuperarían casos que realmente no son la solución adecuada.

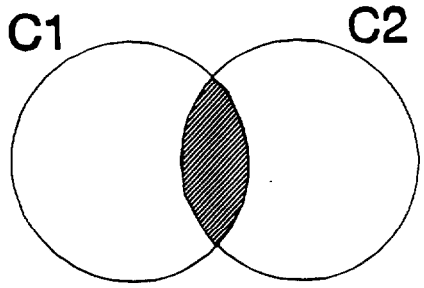


Figura 20. Solape entre dos Casos de memoria

Intuitivamente hablando, el umbral representa el punto a partir del cual se puede garantizar que una determinada descripción de entrada (Ca) es suficientemente similar a alguno de los casos de memoria. Esta definición no es restrictiva por cuanto acepta que más de un caso puedan pasar el corte. Profundizando un poco en este contexto, los descriptores de un caso los podríamos descomponer entre aquella parte que es propia del caso, y la que comparte con otros casos de la base. La figura 20 representa dos posibles casos de memoria mediante dos conjuntos de descriptores con una parte en común. En principio, para que una descripción de entrada Ca de como resultado alguno de esos casos de memoria, tendrá que verificar estas dos restricciones:

(1)

- Que la cantidad de información que describe a Ca es suficientemente representativa
- Que la parte propia del caso Ci que está en la descripción Ca sea distinta del conjunto vacío.

Realmente ambas restricciones se podrían agrupar en sólo una:

(2) - Que la descripción de Ca incluya información significativa de la parte propia de Ci.

En la figura 21 se muestran dos posibilidades de descripción del caso de entrada Ca. En ambas situaciones, si el área que ocupa Ca dentro de la parte propia de C1 es suficientemente grande, diremos que C1 es un caso resultado de la recuperación.

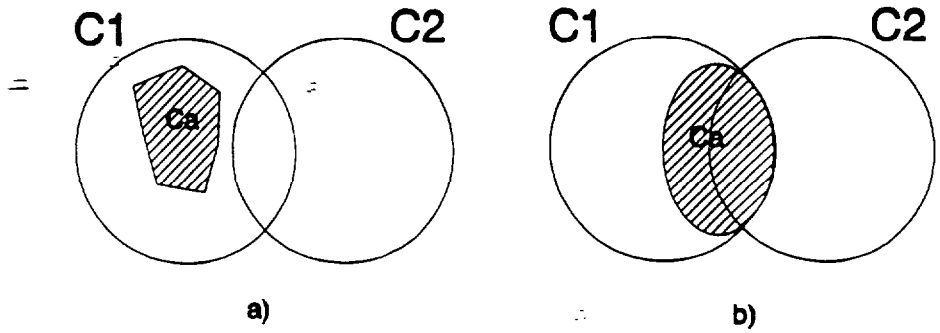


Figura 21. Posiciones de la situación actual respecto a los casos de memoria.

Además de las comentadas anteriormente, otras alternativas para el cálculo del umbral pasan por establecer un mecanismo automático de cálculo. Si calculamos un umbral para cada caso de memoria podríamos tomar como umbral del sistema un umbral medio, o el mínimo, o el máximo. Si usamos el umbral mínimo estaremos tomando una posición de riesgo permitiendo que se alcancen soluciones sin las suficientes garantías de calidad. Un umbral máximo es una posición mucho más conservadora, y uno medio tiene las ventajas (e inconvenientes) de las otras dos opciones. Pero ya que es necesario calcular un umbral para cada Ci, ¿porqué no dejarlos y trabajar con ellos? El único motivo que justificaría un umbral global sería un coste computacional excesivo en la gestión de umbrales individualizados, a la vez que no se produjera una degradación excesiva en el funcionamiento del sistema. Consideramos que el coste de gestión de umbrales particulares para cada Ci no es significativo, y que la degradación del Sistema que pudiera dar lugar la utilización de un umbral global como pudiera ser el medio, va a depender de la desviación típica en del umbral medio. Desviaciones típicas muy altas significa que nos podemos encontrar con umbrales muy pequeños, y otros muy altos, con lo que obtenemos un Sistema que, o es demasiado conservador, o demasiado arriesgado. Si usamos un umbral local a los Ci, el resultado de la recuperación ya no vendrá impuesto por el Ci de mayor valor de FS, sino por aquel que supere significativamente su umbral. De esta forma podremos encontrarnos con

configuraciones del vector solución como la de la tabla 3, en la que el caso con mayor proporción de similaridad no representa realmente una solución a Ca. En este ejemplo, el tercer caso el único que ha superado el criterio para considerarlo como solución. Esta ordenación es factible desde el momento que una descripción de entrada Ca puede incluir factores que pertenecen a más de un caso de memoria. Esta situación se representa gráficamente en la figura 22.

Ci	Valor FS	Umbral
U	83	85
V	82	86
X	80	78
Y	...	...

Tabla 3

La utilización de un umbral particular para cada Ci representa un salto cualitativo en la interpretación de la similaridad entre dos situaciones. El grado de similaridad *prop* que resulta de la aplicación de la función de similaridad FS pasa a un segundo plano. El umbral garantiza que por mucha información que se proporcione a un caso, hasta que la calidad de la información suministrada no garantice su pertenencia a la misma clase el caso Ci no podrá considerarse para su recuperación. Esto se traduce en que muchas veces, hasta que no se proporciona ese concepto 'estrella' o de especial relevancia para el caso de memoria, este no consigue superar su umbral.

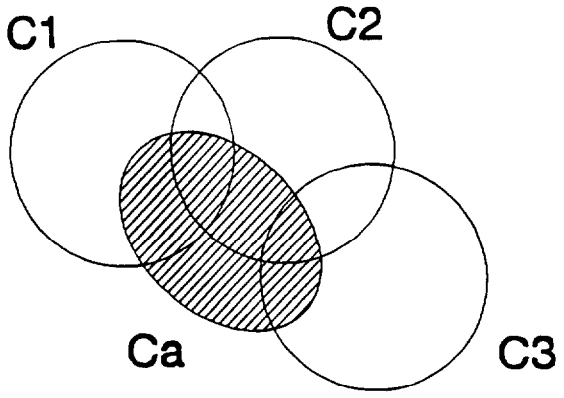


Figura 22. Múltiples solapes entre los casos de memoria y la situación actual.

Pretendemos, mediante la definición de un umbral propio, que cada caso almacenado en memoria tenga sus propios criterios para decidir si la descripción del caso actual Ca le permite distinguirse suficientemente del resto de los casos de la base. Un caso seleccionado deberá posteriormente demostrar que realmente es aplicable a la nueva situación aportando su camino de resolución sin que se quede ninguna meta sin verificar. En definitiva, lo que hemos hecho es dividir el proceso de recuperación en dos estados:

- el primero busca en la base de casos aquellos que poseen características propias en común con la descripción actual
- Para los casos que verifiquen este criterio se intentará adoptar el camino que se siguió cuando se resolvieron

La ordenación del vector solución vendrá determinada por la distancia entre el umbral previsto para el Ci, $\bar{y}$ el valor de similaridad *prop*. En definitiva, los casos ya no compiten unos con otros por compartir más descriptores con Ca, sino contra sus umbrales respectivos. Este hecho nos obligará a realizar una interpretación más elaborada y más realista del vector solución.

3.2 Alternativas al cálculo del Umbral

Primera Aproximación

La primera aproximación al cálculo del umbral trataría de sacar provecho de las dos partes que identificamos anteriormente en la descripción de un caso (1): la propia, y la compartida o común. Se podría decir que la zona en común es la que permite centrar el problema sobre cierta categoría de clases, mientras que la zona propia es la que proporciona la especificidad necesaria para seleccionar una instancia concreta de una clase. Así, el umbral del caso Ci se situaría en el valor que proporciona la zona común con otros casos de la base. Para los casos de la siguiente tabla

C1	C2
	j
b	k
	c
	d

Tabla 4

La zona común se correspondería a la zona sombreada (descriptores c y d). El umbral se situaría en la suma de las aportaciones de c y d. Sin embargo, esta propuesta no suficientemente consistente porque dos zonas comunes pueden 'pesar'[3] distinto para dos casos, como se muestra en la tabla 5.

Pesos[4]	C1	C2	C3
1		j	e
2	b	k	d
3	c	c	j
4	d	d	c

Tabla 5

De esta manera, la zona común del caso C3 pesa menos respecto a C1 que C2. A medida que se van incluyendo más casos en la base, la zona común puede ir variando y creciendo. El umbral para el caso Ci sería la mayor puntuación que obtendría cualquier otro caso de la base teniendo en cuenta sólo la zona común de Ci con el resto de los casos. Para el ejemplo, el umbral de C1 sería 7 ($FS(C2)_{C1} = 7$).

Para se consistente con (2), el método debe garantizar que una descripción de Ca que únicamente utilice descriptores comunes a más de un caso no permita que resulte alguno de ellos ganador. Dados los casos descritos en la tabla 5, esto permitiría resolver situaciones descritas con Ca={b,c,d} para C1, o {j,e,c,} para C3. En el primer caso FS=9, y en el segundo FS=8, lo cual supera en ambos casos el umbral establecido de 7.

Sin embargo, existen una serie de situaciones en la que este método no funciona tan eficientemente. Supongamos ahora dos situaciones como las de las tabla 6. La situación C1 obtiene un umbral de 12 con la suma de las aportaciones de las características que tiene en común con el caso C2 (c,d,e). Podremos garantizar que el caso C1 es un buen caso a recuperar cuando su puntuación para un Ca determinado sea mayor de 12. Por ejemplo, para un consulta Ca={a,b,c,d} FS=14 lo que es mayor que el umbral. Podríamos esperar que para una consulta como Ca={a,b,c,e} también resulte el caso C1 como ganador. Sin embargo no ocurre esto; a pesar de haber proporcionado toda la información característica definida para el caso C1, este no va a ser recuperado porque no supera su umbral (FS(C1)=10). En

[3] Los pesos representan la aportación de cada descriptor a la FS para cada Ci. Por ejemplo, para un Ca={a, b, c}, FS(C1) = 5.

[4] Por simplicidad suponemos que los pesos se aplican por igual a las características de todos los casos. Esto no tiene ni mucho menos que ser así. Cada descriptor de cada caso puede tener un peso específico diferente

principio, esta situación podría justificarse alegando que la característica *d* es especialmente relevante y que no debe omitirse, a la vez que las características propias de C1 no poseen entidad suficiente. Pero con una base de casos con sólo esas dos situaciones ninguno de esos motivos posee la suficiente consistencia.

Pesos	**C1**	**C2**
1	e	j
2	a	k
3	b	d
4	c	c
5	d	e
Umbrales	**12**	**10**

Tabla 6

Tampoco el método permite ser consistente con las afirmaciones de (1). Tomando ahora como referencia a C2 en la tabla anterior. Para un Ca={a,c,d,e} se obtiene una FS(C2)=12, lo cual supera sobradamente su umbral de 10, a pesar de Ca sólo contiene características comunes con C1. Sin embargo, C1 no consigue superar su umbral (FS(C1)=12), y C2 sí. Este hecho nos permite darnos cuenta de una matización a lo apuntado en (1) y, por lo tanto, en (2): esta situación no puede, o no debe considerarse como un error. Desde el momento en que estamos asumiendo que el sistema debe ser capaz de hacer frente a distintos contextos, y de ahí viene el que posibilitemos que una misma característica pueda aportar valores distintos a casos distintos. En función de su relevancia a una situación determinada, una misma característica pesará más o menos. Desde este punto de vista, no es un disparate ni mucho menos, pensar que una característica, o mejor dicho un conjunto de características sean lo suficientemente relevantes para un caso determinado como para que este pueda considerarse para el proceso de recuperación. Esto debe ser así, siempre y cuando el peso de esas mismas características en los otros casos de la base no sea también significativo. En este caso, será necesario buscar los apoyos necesarios para poder decantarnos por alguna de las alternativas. Siendo realistas, situaciones en la que la zona propia de algún caso sea nula como se muestra en la figura 23 pueden ser posibles.

Si bien esto aclara nuestra visión de este último caso particular, aún queda por resolver el primer error que mostrábamos. La característica que queda por incluir en C1, no es una característica distintiva que permita al caso C1 sobresalir. El caso C1 ya sobresale a C2 con la información proporcionada. El descriptor *d* sólo permitirá hacer mayor esta diferencia.

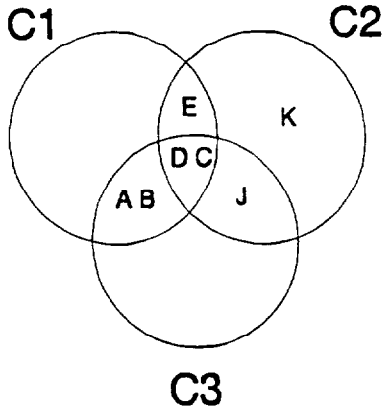


Figura 23. Casos con zona propia nula (C1 y C3).

Segunda Aproximación

Cualquier alternativa que se proponga debe pasar por relajar en la medida de lo posible el método anteriormente comentado. Si asumimos que existen para cada caso de memoria una serie de descriptores que son los que aportan la mayor cantidad de similaridad a los casos, podemos hacer que sean estas características las que controlen el cálculo del umbral. El método consiste en fijar el umbral al nivel de las principales características del caso en cuestión que hace que otro caso de la base le iguale o supere. El algoritmo se muestra en la tabla 7, y en la tabla 8 se muestra una traza de este método.

```
Hacer Ca= Ck para el que se quiere calcular el umbral
Calcular FS(Ci)  para todo i
Mientras     max FS(Ci) corresponda al CK y
             queden características por eliminar
        Eliminar de Ck la característica que menos aporta a FS
Fijar el umbral de Ci al último max FS calculado
```

Tabla 7

Pesos	**C1**	**C2**
1 2 3 4 5	a b c d e	j d c k e
Ca = C1	15	10
Ca = C1-{a}	14	10
Ca = C1-{a,b}	12	10
Ca = C1-{a,b,c}	9	7
Ca = C1-{a,b,c,d}	5	5
Umbral de C1	**5**	

Tabla 8

En este ejemplo observamos como el valor de la FS va disminuyendo progresivamente en ambas situaciones hasta llegar un punto en el que convergen. Esto se debe a que tienen características importantes en común. Sin duda se mejora el comportamiento frente al método anterior al obtener un umbral más bajo. Con aquel método, el umbral de C1 hubiera sido de 10 (la zona en común). Este umbral le habría incapacitado para ser solución frente a entradas del tipo Ca={a,b,c,d} (FS=10). En cambio, con el nuevo umbral se hace mucho más asequible la recuperación de este caso. Es interesante hacer notar que, en línea con lo comentado anteriormente, existen una serie de combinaciones que hacen al caso C1 solución aunque se compongan únicamente de descriptores compartidos (por ejemplo: {d,e}).

Este método impone una serie de restricciones sobre el esquema representacional del sistema que no lo hace aplicable a cualquier sistema CBR:

a- Se parte de una base de conocimiento completa. En el sentido de que cualquier modificación del contenido de alguna de las estructuras obligará a recalcular todos los umbrales.

b- Los casos que componen la base son casos independientes; es decir, no se presupone que la recuperación va a implicar a una combinación de casos, aunque esto sea posible. Deben estar estructurados de forma que esto sea posible (que no existan casos que están incluidos en otros, o que sean iguales)

c- Consecuencia de lo anterior, cada caso debe poseer al menos una característica distintiva respecto a cada uno de los otros casos de la base, aunque no la misma con todos necesariamente.

d- Es posible realizar una normalización de los contenidos de los casos, es decir, una ordenación de las aportaciones de cada descriptor del caso.

Las condiciones a y d son computacionalmente muy costosas, aunque el trabajo computacional se puede realizar durante la fase de entrenamiento del Sistema. Entonces, aunque el cálculo del umbral sea costoso y no se pueda realizar en tiempo real, se pueden adoptar políticas de duplicidad de bases, o de trabajo en batch. Con la primera política el Sistema proporciona una versión de la base de conocimiento a los usuarios, la cual está totalmente verificada y validada. Cuando el Sistema aprende algo nuevo, y precise de una reorganización o de cualquier cosa que implique el cálculo de nuevos umbrales, el Sistema trabaja con una copia de la Base de Conocimiento y mientras los cálculos se realizan, los usuarios siguen trabajando con la copia anterior. Una vez los cambios han sido asimilados, la nueva base de conocimiento se pone a dispocisión de los usuarios. La segunda opción implica la utilización de los tiempos muertos (horas en la que nadie utiliza el sistema) para reorganizar la base de conocimiento. En cualquier caso, ambas políticas no son fáciles de implementar desde el momento en el que se puede 'aprender' más rápido que lo que se tarda en recalcular los nuevos umbrales.

Para hacernos una idea del coste que entraña el cálculo del umbral, vamos a hacer una estimación del número de consultas que serían preciso realizar para una base con 50 casos, donde cada caso viene descrito por una media de 30 descriptores. Para estas estimaciones utilizaremos métodos no optimizados.

La normalización consiste en ordenar las aportaciones de cada característica al caso. Serán necesarias tantas pruebas como características para determinar cual de ellas es la que menos aporta a la solución. Una vez eliminada la que menos aporta se vuelve a ejecutar el sistema otras tantas veces como características iniciales menos una. Así podremos eliminar la segunda. El proceso debe continuar tantas veces como características queramos ordenar. No es posible saber con una única ejecución qué valor aporta cada descriptor al total porque el orden en el que se introduce un descriptor en el caso influye en el valor que este puede conseguir. Lo que permanece invariable es la suma total de aportaciones. Esta no depende del orden en el que se define la información. Para el caso C1 de la tabla 8, serían necesarias las ejecuciones que se muestran en la tabla siguiente:

Descriptores	**nº de consultas**
a b c d e	5
b c d e	4
c d e	3
d e	2

Tabla 9

Esta serie de números se corresponde con la suma de los primeros n términos de una sucesión de razón 1, cuyo término general es:

(3) $Sum = n*(n+1) / 2$

Para el ejemplo con el que estamos trabajando, teniendo en cuenta que la última ejecución no es necesaria, y que la base se compone de 50 casos:

Sum-normalización= ((30*31/2)-1)*50= 23.200 ejecuciones

Para calcular el umbral el proceso es casi el mismo. La única diferencia está en que podemos suponer que para cada caso no tendremos que llegar al número máximo de ejecuciones. Un número razonable de intentos se situaría en la mitad de características por término medio. Es decir, al termino general de (3) debemos restarle los n/2 primeros términos, con lo que el número de ejecuciones es:

Sum-Umbralización= ((((30*31)/2) - ((15*16)/2)))* 50 = 17.250 ejecuciones

Si ahora suponemos que cada ejecución supone 1sg. de cpu (tiempo mucho más que razonable), nos saldría un tiempo de ejecución de aproximadamente 11 horas. Nos interesa resaltar que las estimaciones que realizamos las hacemos poniéndonos en el peor caso. Por ejemplo, la ejecución puede tardar 1sg. si se evalúan todos los casos de la base. Suponemos que el algoritmo utiliza el mismo código que el de una consulta normal, por lo que cualquier dato de entrada se refleja sobre la puntuación de cada caso de la base. Especializar este algoritmo supondría una importante reducción en el tiempo de ejecución. De igual forma, una media de umbralización que necesite la mitad de las características del caso es una predicción muy pesimista. Nuestras pruebas indican que el solape de características entre los casos es bastante mayor. En definitiva, consideramos que este método de cálculo del umbral es un problema abordable, y asumible para cualquier base de conocimiento, siempre y cuando no se le exija una respuesta en tiempo real. Entendiendo por tiempo real lo que tardaría en llegar otra consulta al sistema.

Cuando utilizamos alguno de los métodos descritos para calcular el umbral debemos ser conscientes del riesgo que estamos asumiendo. Dada una determinada descripción de entrada Ca, evaluar si existe algún caso con suficiente entidad para ser recuperado es un problema que depende del Ca en particular con el que estamos tratando, y no exclusivamente de la configuración de la base de conocimiento. La discusión acerca de la zona propia y zona común entre los casos de la base realmente carece de sentido si no se particulariza para una descripción determinada. Como se mostró en la figura 23, lo más probable es que ningún caso

de la base tenga una zona propia absoluta. Un umbral que no se calcule teniendo en cuenta este aspecto debe ser considerado, en el mejor de los casos como una aproximación. Y como sucede con todas las aproximaciones, existirán situaciones en las que la medida falla. Para el cálculo del umbral nosotros hemos partido de una premisa: que las similaridades relevantes entre dos situaciones se encuentran en los niveles de mayor aporte a la función de similaridad, y que las similaridades a niveles inferiores son irrelevantes. Ahora tenemos que estudiar si las situaciones en las que esta aproximación falla son asumibles.

Pesos	**C1**	**C2**	**C3**	**C4**	**C5**
1	c	c	a	j	a
2	f	a	b	k	f
3	a	f	c	e	g
4	g	g	d	d	h
5	k	e	e	l	b
Cálculo Directo	0	5	5	0	0
MZC	10	10	8	9	9

Tabla 10

Partiendo de las situaciones presentadas en la tabla 10, obtenemos los umbrales que se muestran en dicha tabla. Comparándolos con el método de la máxima zona en común (figura 24), ahora son sensiblemente menores. Resaltan especialmente los umbrales cero de los casos C1, C4, y C5. Un umbral cero significa que cualquier dato que coincida con la descripción del caso constituye una señal suficiente para recomendar la recuperación del mismo. Esto se puede interpretar como un reconocimiento a los casos que tienen características distintivas en los niveles de mayor aportación como la *k* para C1, la *l* para C4, o la *b* para C5. Sin embargo, el método descuida las aportaciones de poco peso, pudiéndose dar situaciones como las que se muestran a continuación en las que descriptores que para un caso tienen muy poco peso, y que incluso aportan más información en otras situaciones, son suficientes para recuperar dicho caso, a la vez que insuficientes para recuperar el caso donde su aportación es mayor:

Ca={e} => FS(C4)= 3, por lo que supera su umbral, y a la vez FS(C3)=5 pero C3 no supera su umbral

Ca={f} => FS(C5)= 2, por lo que supera su umbral, y a la vez FS(C2)=3 pero C2 no supera su umbral

También se puede dar una circunstancia distinta, y es que una misma combinación facilite la recuperación de más de uno de los casos de la base, como por ejemplo:

Ca={c,f,g} => FS(C1)= 7, por lo que supera su umbral, y a la vez FS(C2)=8 que también supera su umbral

Esto no representa una situación de error necesariamente. Sólo indica que esos descriptores tienen un nivel de relevancia aceptable para esos dos casos. Una determinada descripción de un caso debe ser capaz de recuperar todos aquellos casos que puntúan desde un punto hacia arriba. Este punto lo define el caso que verifique la prueba de recuperación con el menor umbral. En estos ejemplos, para la primera situación el C4, para la segunda C5, y para la tercera prueba C1.

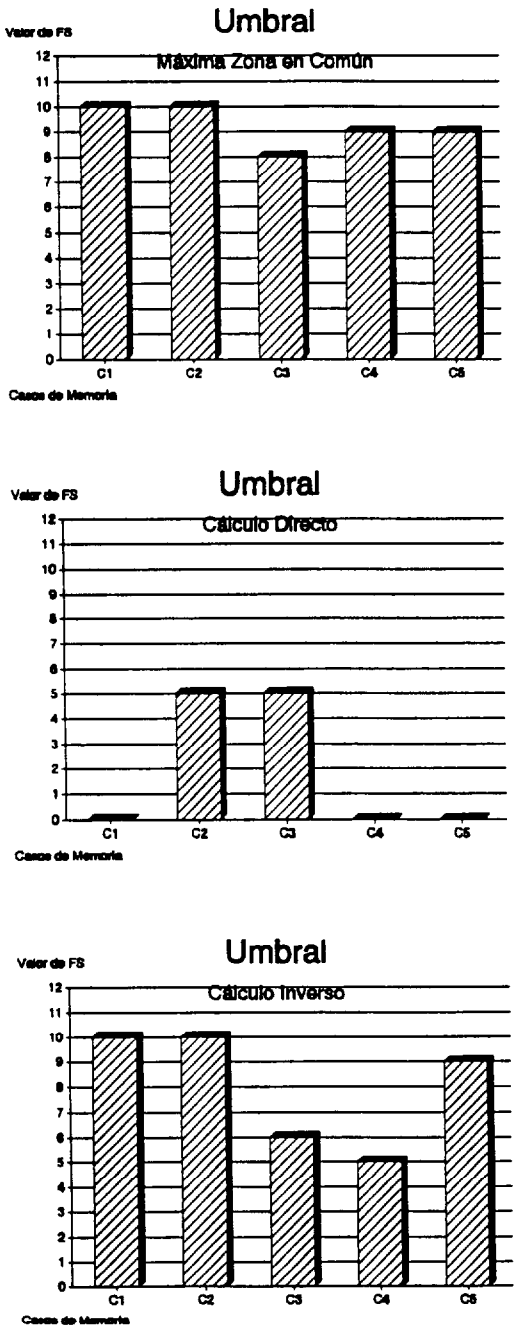


Figura 24. Resultados para distintas posibilidades de cálculo del umbral.

La debilidad de métodos como este, no puede superarse ni siquiera mediante la combinación con otras variantes del método, como puede ser empezar a descartar desde el descriptor más significativo al de menos peso. Si bien, como muestra la figura 24 los umbrales se acercan más a los del método inicial (máxima zona en común), el método sigue teniendo algunas de las mismas deficiencias de la otra versión: se pueden dar umbrales cero (tabla 11). Además, a medida que se realizan más combinaciones de distintas variantes del método (empezar por características de peso intermedio), los umbrales se aproximan más a los del método mzc, dejando a un lado la complejidad computacional, que entonces sí que sería inabordable. Al acercarse los pesos a los del mzc, nos damos cuenta que una de las razones que nos motivaron a buscar métodos alternativos de búsqueda del umbral deja de verificarse. Por ejemplo la situación que se ilustraba mediante la tabla 6 vuelve a reproducirse.

Pesos	**C1**	**C2**
1 2 3 4 5	a b c d e	c b f g e
Ca = C1	15	8
Ca = C1-{e}	10	3
Ca = C1-{d,e}	6	3
Ca = C1-{c,d,e}	3	2
Ca = C1-{b,c,d,e}	1	0
Umbral de C1	**0**	

Tabla 11

La principal conclusión que sacamos de este estudio es que no es posible determinar si un caso de memoria puede ser un buen candidato a la recuperación, sin realizar un estudio del contexto en el que se desarrolla la recuperación. Esto es, como influye una descripción Ca en concreto sobre cada uno de los casos de la base. Lo que viene a significar tomar una decisión para cada caso de memoria para cada combinación posible de descriptores de entrada. Esto evidentemente no es tabulable (hay tantas combinaciones distintas como el factorial del número de descriptores distintos), y si lo fuera entonces no tendríamos problema que resolver, y por lo tanto, debe realizarse en tiempo de ejecución.

Tercera Aproximación

El método que fije el umbral debe buscar el punto a partir del cual se garantiza una convergencia hacia el representante de la clase Ci. Es decir, que cualquier adición en la descripción del caso Ca sólo sirva para consolidar la creencia de que el caso Ci es un candidato a ser recuperado. Esto viene a significar, que si ampliamos la información que describe a Ca con la información adecuada, la proporción de similaridad del caso Ci aumenta. En el peor de los casos, la adición de nueva información en la descripción del caso Ca no variará la proporción de Ci (no puede disminuir).

Retomando el problema en su origen, dadas las dos situaciones de la tabla 6, para un Ca={a,b,c,d} el caso C1 debe ser claramente un caso ganador. Formulándolo de forma un poco más abstracta, el caso Ci deberá ser recuperado cuando la información que le queda por introducir en Ca no puede dar lugar a ninguna ambigüedad respecto a Ci. Los descriptores que pueden ser conflictivos son los que se comparten con otro caso de la base y que todavía no pertenecen a la descripción de Ca. En nuestro ejemplo, el conjunto de descriptores conflictivos vendría dado por *{e}*. Para un Ca distinto, como {a,c}, el conjunto conflictivo de C1 respecto a C2 para Ca sería *{d}*. Expresando esta idea mediante una inecuación:

$$(4) \qquad FS(Ci)_{Ca} + (FS(Ci)_{Cj} - FS(Ci)_{Cj \cap Ca}) > FS(Cj)_{Ca \cap Ci} + (FS(Cj)_{Ci} - FS(Cj)_{Ci \cap Ca})$$

Donde:

- $FS(Ci)_{Ca}$: es la puntuación que obtiene Ci respecto a Ca
- $FS(Ci)_{Cj}$: es la puntuación que obtiene Ci cuando la entrada en Cj
- $FS(Ci)_{Cj \cap Ca}$: es la puntuación que obtiene Ci cuando la entrada es la parte común entre Cj y Ca
- $FS(Cj)_{Ca \cap Ci}$: es la puntuación que obtiene Cj cuando la entrada es la parte común entre Ci y Ca. (equivalente a $FS(Cj)_{Ci \cap Ca}$)
- $FS(Cj)_{Ci}$: es la puntuación que obtiene Cj cuando la entrada es Ci

La anterior expresión puede simplificarse:

$$(5) \qquad FS(Ci)_{Ca} + (FS(Ci)_{Cj} - FS(Ci)_{Cj \cap Ca}) > FS(Cj)_{Ci}$$

Lo cual se puede leer como: la puntuación que obtiene Ci más lo que le falta de la zona común con Cj, debe ser mayor que la puntuación que obtiene Cj por la zona común con Ci. Es decir, Ci debe verificar que los descriptores que tiene en común con Cj que todavía no pertenecen a la descripción de Ca, van a aportar suficiente información como para impedir que la puntuación que obtenga Cj con esos descriptores no sobrepase a la de Ci.

Esta fórmula garantiza un funcionamiento correcto incluso con situaciones como la de la figura 23, con Ca que incluyan al menos alguna característica distintiva. Por ejemplo, con

Ca={a,c} la fórmula sería:

$$4 + (12 - 3) > 12$$

con lo que el caso C1 podría ser seleccionado.

La siguiente tabla muestra el funcionamiento de esta fórmula para algunas situaciones de la tabla 10.

Ca	Caso Base	Caso Competidor	Fórmula	Respuesta
a, b	C3	C5	3+(3-3)>6	Rechazado
a, b	C3	C4	3+(9-0)>9	Seleccionado
c, d	C3	C5	7+(3-0)>6	Seleccionado
a, e	C3	C2	6+(9-6)>8	Seleccionado
a, e	C3	C4	6+(9-5)>7	Seleccionado

Tabla 12

En (5) lo que se representa realmente es una proyección. Indica cual puede ser el valor final de la función de similaridad una vez se haya introducido el resto de descriptores en común con el caso competidor. Por lo tanto, no es por sí sola una prueba de recuperación. No sirve para decidir si con el estado actual de Ca el caso Ci es ya solución. Tal vez pueda apreciarse mejor este hecho con un ejemplo. En la tabla 12 aparece en la fila correspondiente al caso competidor C2 una respuesta de Seleccionado. En el campo 'Fórmula' se utiliza la representación de (5). Si en lugar de esta utilizamos la de (4), tendríamos:

$$\mathbf{6} + (9 - 6) > \mathbf{7} + (8 - 7)$$

Donde los números resaltados indican cual es la respuesta de los casos C3, y C2 para una entrada de Ca, y Ca$\cap$C3 respectivamente. Es decir, que actualmente es más similar el caso C2 que el C3 para una entrada de Ca={a, e}. La proyección por sí misma no es capaz de detectar este hecho. Podríamos ilustrarlo gráficamente en la figura 25. En ella observamos que actualmente es C3 quien está por debajo de C2, aunque la proyección nos indica que C3 será quien *puede* acabar por encima, aunque eso depende de como se complete la descripción de Ca. La situación actual se podría interpretar como una situación de 'información insuficiente'; sería necesario enriquecer Ca para decidir si seleccionar C3, u otro caso que cumpla con las restricciones. En definitiva, necesitamos complementar (5) con una prueba más:

(6) $$FS(Ci)_{Ca} > FS(Cj)_{Ci \cap Ca}$$ para todo j <> i.

O sea, que la puntuación que obtiene Ci debe ser la mayor considerando sólo la parte de Ca que está en Ci.

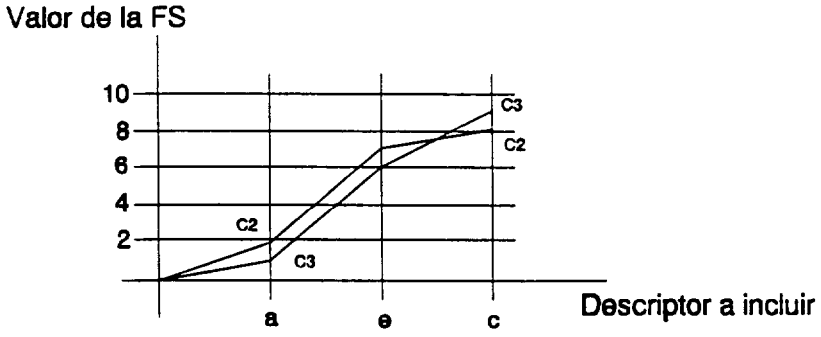


Figura 25. Evolución en la puntuación de los casos. La superioridad de un caso determinado sobre otro, puede ser sólo temporal.

Lo expuesto en (5) debe ser revisado para incluir las situaciones en las que la función de similaridad proporciona mejores respuestas para un Cj con un subconjunto de la información que aún queda por incluir y que tiene en común con Ci. En la figura 26 mostramos dos situaciones en las que la posible secuencia de entrada de descriptores comunes puede producir sucesivas alternativas en la valoración de la similaridad (FS). Como (4) sólo contempla una situación terminal (cuando se ha introducido *toda* la información en común entre Ci y Cj) tenemos que redefinir dicha inecuación:

(7) $$FS(Ci)_{Ca \cup X} > FS(Cj)_{Ci \cap Ca \cup X}$$ para todo X, j y j <> i.

(8) $$X = \{d_1,..d_n : d_i \in D,\ d_i \notin Ca,\ d_i \in Ci \cap Cj\}$$

donde D es el dominio de descriptores del sistema
d_i es un descriptor concreto

Esta expresión comprueba que para todos los conjuntos que se pueden formar con la parte en común entre Ci y Cj que no pertenece a Ca, la evaluación de la función de similaridad para Ci es mayor que para Cj.

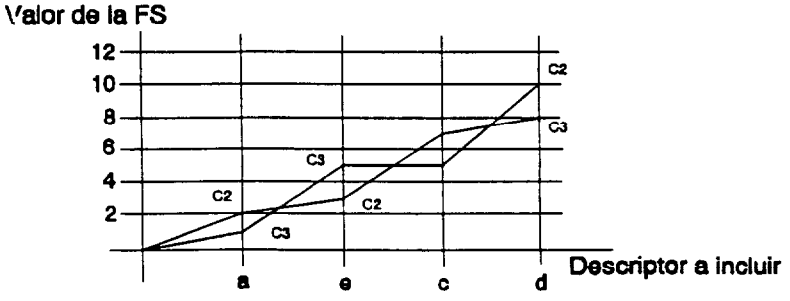


Figura 26. Evolución de la puntuación de dos casos con múltiples alternativas durante la inclusión de los descriptores.

Ahora estudiemos la situación que se produce cuando utilizamos los casos de la tabla 13. Para un Ca={a} el caso a recuperar será el C4, porque verifica tanto (6) como (7). Sin embargo, ¿tiene sentido que recuperemos una situación con una evidencia tan pobre como parece ser esa? ¿porqué se descarta al resto de los casos cuando la diferencia entre todos es mínima?. El concepto que se está agrediendo en este ejemplo es el de la relevancia de la información. ¿Cuándo podemos decir que una cierta cantidad de información que describe una situación es significativa?. Nosotros hemos introducido el *Umbral Mínimo Significativo* (UMS). Este umbral marca el punto a partir del cual podemos decir que la puntuación que obtiene un caso Ci puede ser considerada como significativa

Pesos	C1	C2	C3	C4
1	a	a	a	b
2	c	d	e	a
3	b	j	k	l
4	...	...	...	...
.				

Tabla 13

Existe una diferencia cualitativa entre lo que significa el UMS y el umbral que hemos estado comentando a lo largo de esta sección. Aquel se utilizaba para decidir si el caso Ci era solución o no. Este nos indica si la cantidad de información que actualmente describe a Ci es significativa. Utilizando un símil con la teoría de los Factores de Certeza, una función se podía considerar verdadera cuando la confianza sobre la veracidad de dicha función superaba el 20% del valor máximo. Por ejemplo, para la evaluación de la función igual (=) se dice que A = 'casa' es verdadera, si el factor de certeza con el que el objeto A tiene asociado ese valor es mayor del 20%.

Al igual que sucede con los factores de certeza, este valor debe ser fijado por el sistema (o por el ingeniero del conocimiento o el experto) y no es posible determinarlo automáticamente. Es un valor que depende de la aplicación concreta que se está desarrollando y con un alto contenido de 'sentido común'. A nosotros lo único que se nos ocurre es que el 20% de los factores de certeza podría ser una buena aproximación inicial al UMS. Este valor no puede ser demasiado bajo para evitar situaciones como la anteriormente descrita, y para evitar tener que comprobar las otras dos condiciones con demasiados casos, lo cual como veremos, tiene un coste computacional alto. Tampoco debe ser demasiado alto, ya que perdemos la ventaja de las otras condiciones: anticiparse al umbral máximo en común. En definitiva, esta última condición se podría especificar como:

(9) $$FS(Ci)_{Ca} > UMS$$

La siguiente tabla resume las condiciones que debe cumplir un caso de memoria para que este pueda ser recuperado. En ella hemos agrupado las expresiones de (6) y (7) al incluir el conjunto vacío en la definición de X:

(10) $FS(Ci)_{Ca \cup X} > FS(Cj)_{Ci \cap Ca \cup X}$ $X = \{d_1,..d_n: d_i \in D, d_i \notin Ca, d_i \in Ci \cap Cj\} \cup \{\emptyset\}$	Ci no va a ser superado con la información que comparte con los otros casos de la base, y es el mejor representante con esa información
$FS(Ci)_{Ca} > UMS$	La parte de la descripción de Ca que tiene Ci es significativa

Tabla 14

Costes

Si bien el análisis de la segunda expresión es trivial (9), no lo es tanto el estudio de la primera (10). Para la expresión (9) no es necesario realizar ningún cálculo adicional sobre la base de conocimiento; sólo hay que ejecutar la comparación y preseleccionar el conjunto de casos de memoria que la verifiquen.

Mucho más complejo es el análisis de la expresión (10). Descomponiendo las dificultades, tenemos que calcular los posibles $Ci \cap Ca$, y $Ck \cup X$:

$Ck \cup X$: La unión entre dos conjuntos de descriptores debe tener en cuenta que a pesar de que un descriptor se define como una tripla de valores, a estos efectos dos descriptores serán iguales si nombran al mismo concepto y atributo. La re-evaluación de la FS para el nuevo conjunto no puede limitarse a una suma de las aportaciones de los nuevos componentes, ya que es necesario descontar primero las aportaciones que recibían por similaridad alternativa cada uno de ellos (por ejemplo, la ausencia de un descriptor d_i era parcialmente cubierta por otro descriptor d_j que sí pertenecía al conjunto de *Ck*). Por otro lado, las particularidades del Sistema que estamos proponiendo posibilitan comportamientos como el de la figura 26, lo que significa que es necesario *calcular y evaluar* cada uno de los posibles conjuntos *X*. El coste del cálculo del conjunto vendrá determinado por el número de descriptores que verifican la definición de X. El número de conjuntos que se pueden construir a partir de X se calcula como:

$$\sum_{n=0}^{k} \frac{k!}{n!\,(k-n)\,!}$$

donde k es el número de descriptores del conjunto X.

En principio, para cada conjunto de X es necesario re-evaluar la función de similaridad (FS), ya que no podemos predecir la aportación combinada de un conjunto de descriptores a partir de las puntuaciones individuales (individualmente pueden ser mayores para el caso Ci, pero combinadamente ser mayores para el caso Cj). Para bases de casos grandes, y para casos descritos a su vez con un gran conjunto de descriptores esta prueba puede degradar sensiblemente el sistema si no se disponen de los recursos hardware adecuados, o el código no está suficientemente optimizado.

Ci∩Ca: El cálculo de la intersección entre dos conjuntos de descriptores en más complejo y más costoso. Aparentemente no habría ningún problema; sólo sería cuestión de marcar todo aquel descriptor en Ca que aporta información a Ci. Sin embargo, tampoco esto es tan fácil como parece. Desde el momento en que el Sistema es capaz de trabajar con Similaridad Alternativa, estamos permitiendo que una misma unidad de 'conocimiento del dominio' pueda ser inferida por más de un camino distinto. O lo que es lo mismo: podremos tener varios subconjuntos de Ca que aportan la misma cantidad de información a Ci, y sin embargo, distinta para un Cj cualquiera de la base. La figura 27 ilustra esta posible circunstancia. Para garantizar que la expresión (10) se verifica respecto a todos los casos es necesario calcular el subconjunto *Ci∩Ca* que más aporta para cada uno de los Cj de la base.

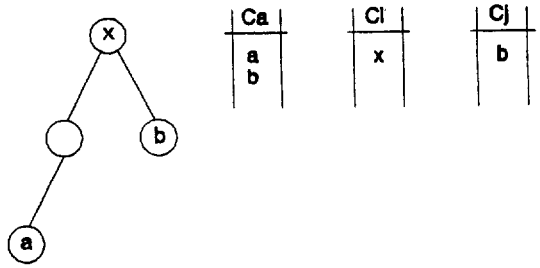


Figura 27. X puede ser parcialmente reconocido por 'a' o por 'b'. Desde el momento en que uno de estos descriptores es introducido (por ejemplo 'a'), el otro es irrelevante para Ci, porque 'a' ya proporciona a Ci la parte proporcional de similaridad respecto a X. Sin embargo, no es lo mismo seleccionar un $Ci \cap Ca = \{a\}$ que un $Ci \cap Ca = \{b\}$, ya que el efecto sobre Cj es distinto en cada caso.

Consideramos que el desarrollo de unos algoritmos eficientes para la verificación de la expresión 10 con un sistema como el que estamos definiendo no es ni mucho menos una tarea trivial, y que necesitará posiblemente de la utilización de técnicas paralelas o de una máquina con la potencia de cálculo necesaria. Hoy por hoy imponen unas restricciones computacionales excesivas para esperar unos tiempos de respuesta aceptables con unos recursos escasos como los que disponemos. Este hecho, sin embargo, no quita validez a las expresiones desarrolladas sino que propone una línea de trabajo a desarrollar.

Aplicabilidad a otros Sistemas de Scoring

Las expresiones de la tabla 14 son válidas para cualquier sistema que utilice una función de scoring para evaluar la similaridad, siempre y cuando no defina un peso global para cada atributo en el dominio. En este último caso, las condiciones desarrolladas en 6, 7 y 9 son irrelevantes o carecen de sentido. El cálculo de $Ci \cap Ca$ es trivial porque cada atributo posee un valor constante (suponiendo que no existe similaridad alternativa y/o otras estructuras de apoyo como una jerarquía de conceptos, o un modelo causal, etc.). Esto elimina la necesidad de (6). (7) más que ser irrelevante es perniciosa porque impediría que el sistema encuentre más de una solución, y (9) carece directamente de sentido ya que como única condición facilitaría demasiadas soluciones.

Sin embargo, manteniendo la restricción de que el sistema carezca de 'similaridad alternativa', y permitiendo definir un valor para cada atributo por cada caso de memoria las expresiones sí que mantienen su validez. El cálculo de $Ci \cap Ca$ es sencillo porque cada descriptor sólo puede tener (o dar lugar a) un único valor por caso; o dicho de otro modo, no es posible deducir una unidad de información por más de un camino. Por lo tanto la reevaluación de la función de similaridad será también un mecanismo sencillo de implementar (posiblemente sólo mediante sumas al valor antiguo de FS para cada Ck).

En conclusión, la característica de similaridad alternativa y el oportunismo a la hora de evaluar el contexto de cada característica son las que complican la evaluación de las expresiones de la tabla 14.

3.3 Respuestas del Sistema

El Sistema proporcionará tantas respuestas como situaciones pueda recuperar y adaptar sus caminos de resolución. Es decir, la definición del Sistema no limita a una el número de posibles soluciones que puede proporcionar frente a una descripción dada. Sin embargo, el

coste computacional será mayor en función del número de soluciones que se quieran buscar. Proponemos que sea un parámetro del sistema a definir por el usuario el que fije el número de soluciones a buscar (una, n, o todas las posibles).

En el capítulo anterior dividimos el proceso de resolución del Sistema en dos etapas:

- la selección de casos candidatos (uno o varios), y
- la adaptación del camino de resolución de estos casos.

Sólo cuando se haya completado con éxito la adaptación de un camino de resolución, el Sistema proporcionará una respuesta.

La selección se realiza de acuerdo a los criterios desarrollados en el apartado anterior. Este proceso puede dar lugar a la recuperación de más de una situación de memoria. En principio todos estos casos pueden aportar su solución a la situación actual. Sin embargo, si varios casos seleccionados pertenecen a la misma categoría de solución (p.e. para un sistema de diagnosis proporcionan la misma enfermedad) tan sólo será necesario adaptar el camino de resolución de uno de ellos.

La adaptación del camino de resolución utiliza las pistas almacenadas en los casos en forma de metas. Estas metas se corresponden con hipótesis parciales que se fueron evaluando durante la resolución del caso de memoria. La evaluación de dichas hipótesis pudo ser exitosa, o no. En el caso de que tuviera éxito también se almacena su justificación causal. El orden en el que se almacenan es significativo por cuanto permiten reconstruir el camino de resolución de los casos de memoria. La adaptación intentará verificar las mimas hipótesis que fueron verificadas para el caso de memoria, y preferentemente utilizando los mismos argumentos que en dicho caso, aunque esto último no es imprescindible. Cuando una meta no se puede verificar en la situación actual con los mismos argumentos que con el caso de memoria que se utiliza como guía en la adaptación, se busca otra alternativa en otro caso de memoria, o a último remedio en la red causal.

La selección de una alternativa debe tener en cuenta el casó de memoria que indirectamente se beneficiará de esa medida mediante un incremento en su puntuación de similaridad debido a que existe un nuevo dato que comparte con la situación actual, y al menos una relación causal más (la que relaciona la justificación con la meta). Este posible incremento puede dar lugar a dos situaciones dependiendo del estado del caso afectado:

a- Si el caso afectado es un caso que ya fue recuperado (que verifica las condiciones de la tabla 14), entonces será necesario reevaluar la FS para comprobar si su

nueva puntuación supera a la del caso guía actual. Si estamos en una situación en la que sólo nos interesa la mejor solución entonces este hecho puede cambiar el centro de atención de la adaptación hacia este caso.

b- Si el caso no verificaba aún los criterios de recuperación, puede que ahora sí este en condición de cumplirlos.

La imposibilidad de justificar una meta del caso guía sobre el caso actual conduce inevitablemente a el abandono del caso guía y de cualquier otro que incluya la verificación de dicha meta en su descripción. Si existe algún otro caso que verifica los criterios de recuperación, se utilizará el mismo como caso guía; sino, será necesario un estudio del estado actual de los casos de memoria para buscar un posible candidato a ser recuperado y utilizado como caso guía.

3.4 Guía al proceso de recuperación

Cuando el estado actual de ejecución del Sistema no es capaz de proporcionar ninguna solución con los datos que inicialmente proporcionó el usuario, es necesario enriquecer el conjunto de descriptores de Ca mediante un interrogatorio selectivo al usuario. Con 'interrogatorio selectivo' queremos decir que el proceso de recabar más información del usuario (o de cualquier dispositivo externo) no debe realizarse aleatoriamente, o dejando una excesiva libertad de selección al usuario, sino que es necesario guiar este proceso proponiendo aquellos descriptores que puedan servir para recuperar un caso, y que a la vez tienen ciertas posibilidades de ser aceptados por el usuario. En definitiva, podríamos plantear el problema como:

Para un caso Ci

Encontrar un $Z = \{d_1,..d_n: d_i \in D, d_i \notin Ca, d_i \in Ci\}$

que verifique

$$FS(Ci)_{Ca \cup Z \cup X} > FS(Cj)_{(Ci \cap (Ca \cup Z) \cup X} >= UMS \text{ para todo } j$$

donde

$$X = \{d_1,..d_n: d_i \in D, d_i \notin Ca \cup Z, d_i \in Ci \cap Cj\} \cup \{\ominus\}$$

o lo que es lo mismo, hemos reformulado las expresiones de la tabla 14 para incluir la definición de Z. Y definimos Z como el complemento necesario para que un caso Ci pueda verificar las condiciones de recuperación.

Podría darse el caso de no encontrar un Z válido para un Ci (o para todos los Ci), ya que Z sólo puede estar constituido por descriptores que no pertenezcan al caso actual (Ca)

(considerando únicamente la pareja del descriptor formada por el concepto y el atributo) y este conjunto puede ser nulo. Por otro lado, sería posible encontrar varios Z que hagan que un caso Ci pueda ser recuperado. Sin embargo, a nosotros tan sólo nos interesa uno de ellos: el que represente un mayor ahorro cognitivo, y que sea el más factible. Sin embargo estas características no tienen porqué venir ligadas. El conjunto Z de mayor ahorro cognitivo será el de cardinalidad menor; el que tenga menos descriptores que verificar. El más factible se compone de descriptores para los que se puede prever de alguna manera que el usuario conoce esa información y además presenta las características adecuadas para el caso Ci (p.e. no basta con responder a la información que se pregunta, sino que la respuesta debe ser la que se espera para Ci). Entonces habría que articular una serie de criterios de preferencia (heurísticos) como podrían ser los siguientes:

- Preferir descriptores de Ci que nombren un concepto que ya pertenece a Ca
- Descriptores que participen en el consecuente de una relación causal marcada como D6, ya que pueden sugerir prolongaciones del camino de resolución.
- En general, preferir descriptores que activen, o permitan activar relaciones causales en común con Ci.
- Descriptores que se utilicen en las justificaciones de las metas del caso Ci, ya que en cualquier caso habría que conocer esta información si el caso Ci es recuperado.

Todas estas heurísticas no sólo tienen el objetivo de simplificar el espacio de estados de Z, sino también de mostrar un comportamiento coherente (¿e inteligente?) de cara al supuesto usuario.

El proceso de recuperación terminará cuando ya no sea capaz de proponer ningún caso más para su adaptación, o cuando se haya conseguido adaptar al menos un caso (dependiendo de la modalidad de respuesta seleccionada). Y no se podrán proponer más casos cuando estos contienen conjuntos Z nulos, o metas que no son verificables para la situación actual.

4 Ajuste de la función de Similaridad

La función de Similaridad está definida como:

$$FS(C_i)_{Ca} = \sum_{j=1}^{8} P_j W_{ij}$$

Donde Ci es el caso para el que se calcula el valor de la función dada una entrada Ca. Wij son los valores que recibe el caso para cada una de las ocho dimensiones, y Pj son los pesos de ponderación, los cuales determinan la importancia relativa de cada dimensión.

La importancia de cada dimensión viene determinada por la carga semántica que poseen cada uno de los aspectos que se tienen en cuenta cuando se valora la similaridad entre dos situaciones, y, por lo tanto, debería ser un valor independiente de la aplicación. Sin embargo, esto no es del todo cierto. Cada dominio se modela de distinta manera, y utilizando distintas formas de representar el conocimiento. Cuando el balance entre los distintos tipos de conocimiento está equilibrado, entonces los pesos pueden establecerse a partir de la carga semántica de la dimensión a la que representan. Cuando alguno de los esquemas de representación predomina sobre otro, debe quedar reflejado este hecho en el sistema de ponderación, aunque siempre dentro de ciertos límites. Por ejemplo, existen dominios que tienen modelos causales muy ricos y densos, y a la vez organizaciones jerárquicas o incluso conocimiento episódico limitado, y viceversa. El desequilibrio también puede darse dentro de una misma estructura, definiendo, por ejemplo, determinados aspectos del dominio con un mayor detalle que otros. Esta descompensación es un caso particular de la anterior ya que no podemos tener estructuras compensadas si una de ellas esta deformada en origen.

Lo usual es que exista algún tipo de descompensación en la formalización del Dominio de una aplicación, ya que no existe ningún método formal o informal que nos permita realizar unas tareas de adquisición del conocimiento con tanta precisión. Incluso el concepto de *un Dominio con unas estructuras de representación compensadas* es ambiguo por naturaleza y sujeto a interpretación. Podemos establecer a posteriori el grado de compensación de las estructuras observando los valores Wij para cada caso de la base. Cuanto más homogéneos sean, mayor será la homogeneidad y el equilibrio. Volveremos sobre este aspecto posteriormente.

Las descompensaciones intra-estructura son más fáciles de detectar porque las distintas dimensiones de una misma estructura están sujetas a ciertas relaciones que impedirían alcanzar un conjunto de pesos coherente si alguna de ellas no se verificara. La tabla 15

muestra estas relaciones entre dimensiones para cada categoría semántica. No tendría sentido asignar más puntuación a un caso que coincida sólo en <concepto+atributo> que a otro que coincide en <concepto+atributo+valor>. Además de este significado de 'sentido común', estas relaciones garantizan que un caso no pueda superar mediante la 'similaridad alternativa' a otro que posea una 'similaridad directa', como podría ser el caso si se potencian las dimensiones alternativas sobre las directas.

O(concepto) > O(concepto cercano)	pcon1 > pcon2
O(relación causal completa) > O(relación causal incompleta)	pcau < rela1 > rela2 >= rela3
O(concepto+atributo+valor) > O(concepto + atributo)	pcontx1 < pcontx2

Tabla 15: Relaciones entre dimensiones de una misma categoría.
'O' representa la puntuación obtenida por la dimensión que referencia

No es posible establecer relaciones similares entre categorías distintas debido a la naturaleza distinta del conocimiento que codifican.

Idealmente, las aportaciones de cada dimensión deberían ser iguales de forma que no existiera ninguna estructura determinante a la hora de tomar alguna decisión. Sin embargo, las relaciones de la tabla 15 impiden que se pueda alcanzar esta situación hipotética de que todos los pesos tengan un valor unitario. Una excesiva descompensación en alguna de las dimensiones reflejaría una excesiva dependencia del Sistema sobre dicha dimensión. La falta de un dato podría ser determinante y difícil de sustituir mediante la similaridad alternativa. El sistema tendría un comportamiento débil frente a ausencias en la información durante la consulta, o vacíos puntuales en la definición del conocimiento de la aplicación.

4.1 Cálculo de los Pesos de la función de Similaridad

Experimentalmente hemos calculado los *Pj* para pequeños dominios de prueba que hemos implementado, y en principio no hemos encontrado mayor dificultad para fijarlos. Sin embargo, este proceso puede ser mucho más difícil para bases de casos más completas o complejas, recomendándose en estos casos algún método automático que los calcule. Técnicas como los métodos inductivos, redes neuronales, programación dinámica, y de minimización (o maximización) por gradiente podrían ser aplicables. Realizar un estudio serio acerca de qué técnica es la más adecuada para esta situación es una tarea compleja, desde el momento en que son áreas que están evolucionando continuamente, y existen muchísimas variantes para

cada familia de métodos. A priori podíamos descartar los métodos inductivos porque requieren un conjunto de ejemplos suficientemente significativo, usualmente les influye el orden en el que se contemplan los ejemplos, además de que están más orientados a clasificar etiquetas lingüísticas; los métodos de minimización por gradiente buscan un mínimo (o máximo) local, por lo que es necesario dar una buena aproximación inicial; la programación dinámica podría servir para buscar un mínimo absoluto si se pudiera acotar el intervalo de búsqueda (sin sentido en nuestro problema), y las redes neuronales suelen tener un coste computacional 'demasiado' elevado. A pesar de que en principio no consideramos el ajuste automático como una función prioritaria de la propuesta que estamos desarrollando, quisimos comprobar la viabilidad de dicha funcionalidad. Gracias al asesoramiento del Dr. Luis Álvarez de este departamento desarrollamos la propuesta de ajuste por minimización por gradiente que se describe en la siguiente sección.

La realización de un estudio similar con las redes neuronales requería de un análisis y de unos fundamentos muy profundos en el campo neuronal, ya que, como se reconoce en [DAY90], muchas de las decisiones a la hora de plantear la topología y tamaño, de seleccionar el paradigma adecuado y el ajuste de sus parámetros internos, tienen un gran componente heurístico que sólo la experiencia puede proporcionar. Por eso, preferimos apuntar esta tarea como uno de las líneas de expansión más prometedoras del presente trabajo.

Pero antes queremos comentar un aspecto relacionado con el conjunto de casos que se deben utilizar para ajustar los pesos. En principio tendremos tres opciones: utilizar sólo los casos de la base de casos, otros casos de prueba distintos de los de la base, y por último, ambos conjuntos. Los casos de la memoria reforzarán los pesos relacionados con la similaridad directa, descuidando en cierta medida la indirecta, a la cual le dará un valor muy pequeño. Y recíprocamente, es de suponer que si utilizamos casos de prueba no incluidos en la memoria de casos la similaridad indirecta podría ser sobredimensionada sobre la directa. En definitiva pensamos que la mejor opción consiste en construir un conjunto de casos de prueba constituido por los casos de la memoria más algunos más de forma que se pueda conseguir un equilibrio en la ponderación de las dimensiones de similaridad. Evidentemente, el ajuste será más preciso cuanto mayor sea el conjunto de casos que se utilice para el ajuste.

4.2 Minimización por Gradiente

Es posible encontrar una buena solución son un método de hill-climbing si somos capaces de proporcionar una buena aproximación inicial. Y nosotros pensamos que esto es posible, ya que partiendo de las restricciones de la tabla 15 y de que nos interesan pesos lo más homogéneos posible, podríamos encontrar un mínimo local que cumpla con dichos

requisitos. Es más, hasta nos podría interesar más encontrar un mínimo cercano al punto de homogeneidad para que el sistema sea más robusto, que un mínimo global más 'óptimo' con pesos más descompensados. Un mínimo más 'óptimo' es aquel que consigue que la diferencia entre la FS para el caso ganador y el mejor caso de otra categoría sea mayor que para cualquier otra combinación de pesos. De igual forma, podemos definir una distancia mínima (d_{min}) de manera que una solución sólo podrá considerarse como aceptable si el resto de los casos que no son solución se mantienen a dicha distancia del caso ganador. d_{min} sólo tiene sentido durante el cálculo de los Pj, y la finalidad de conseguir un sistema más robusto. Durante la consulta se utilizarán los criterios de la tabla 14 para decidir si se ha alcanzado una solución. El valor de d_{min} debe venir impuesto externamente y teniendo en cuenta las características propias del dominio que se está formalizando. Por ejemplo, dependiendo del solape entre distintas categorías de solución d_{min} podrá ser mayor o menor. A mayor solape, menor puede ser el tamaño de d_{min}, y viceversa.

Dado un caso de entrada Ca definido mediante un conjunto de descriptores $\{c_1, c_2, ..., c_k\}$, su evaluación en el Sistema producirá una respuesta para cada caso de memoria y para cada dimensión, lo cual lo podríamos representar mediante X_j^i, donde la X representa el valor de respuesta para la dimensión j y el caso i frente a la entrada Ca. De esta forma podemos formular el problema de recuperación[5] como:

$$(11) \qquad \frac{\sum_{j=1}^{8} P_j X(Ca)_j^{Co}}{\sum_{j=1}^{8} P_j X(m)_j^{Co}} \geq \frac{\sum_{j=1}^{8} P_j X(Ca)_j^{i}}{\sum_{j=1}^{8} P_j X(m)_j^{i}} + d_{min} \quad \forall i \neq Co$$

Donde *Co* es el caso de memoria que debe ser la solución para la situación actual *Ca*, e *i* identifica al resto de los casos de la base. Por lo que:

$X(Ca)_j^{co}$: indica el valor de la dimensión *j* para el caso de memoria *Co* frente a la entrada *Ca*
$X(m)_j^{co}$: indica el valor de la dimensión *j* para el caso de memoria *Co* frente a la entrada *Co*
$X(Ca)_j^{i}$: indica el valor de la dimensión *j* para el caso de memoria *i* frente a la entrada *Ca*
$X(m)_j^{i}$: indica el valor de la dimensión *j* para el caso de memoria *i* frente a la entrada *i*

Entonces para cada caso de entrada Ca obtenemos *(n-1)* condiciones sobre *{Pj}*.

[5] Para este proceso no tienen sentido las expresiones de la Tabla 14 porque estamos trabajando con las descripciones completas de los casos. Es decir, estamos suministrando toda la información disponible y libre de error.

Si ahora disponemos M casos de evaluación (*n* casos de la base más otros *k* de prueba de los que conocemos su respuesta), tendremos *Mx(n-1)* condiciones sobre *{Pj}*. O sea, que haremos variar *Ca* desde 1 hasta *Mx(n-1)*,

Haciendo $W^{Co} = \sum P_j X(Ca)_j^{Co}$, $W^{Com} = \sum P_j X(m)_j^{Co}$, $W^i = \sum P_j X(Ca)_j^i$, $W^{im} = \sum P_j X(m)_j^i$, podemos transformar (11) en

$$\frac{W_k^{Co}}{W_k^{Com}} - \frac{W_k^i}{W_k^{im}} \geq d_{min} \quad \forall i \neq Co,\ k=1,\ldots,Mx(n-1) \tag{12}$$

Nos interesa que el mayor número de registros cumplan esa propiedad (idealmente todos). Podemos definir una función escalón:

$$\begin{aligned} \tilde{f}(x) &= 1 \quad \forall x \leq 0 \\ \tilde{f}(x) &= 0 \quad \forall x > 0 \end{aligned} \tag{13}$$

Es equivalente a minimizar

$$G(P_1, \ldots, P_8) = \sum_{k=1}^{Mx(n-1)} \tilde{f}\left(\frac{W_k^{Co}}{W_k^{Com}} - \frac{W_k^i}{W_k^{im}} - d_{min}\right) \tag{14}$$

Si cambiamos $\tilde{f}$ por una f derivable, entonces podremos utilizar un método de minimización por gradiente. Tomando f como:

$$f(x) = \frac{Ae^{-x}}{1 + Ae^{-x}}$$

donde A es cualquier constante distinta de cero, y sustituyendo x por (14), y derivando respecto a cada Pj obtendremos:

$$\frac{\partial G(P_1, \ldots, P_8)}{\partial P_q} = \frac{\sum_{k=1}^{Mx(n-1)} -A\, e^{-\left(\frac{W_k^{Co}}{W_k^{Com}} - \frac{W_k^i}{W_k^{im}} - d_{min}\right)}}{\left(1 + A\, e^{-\left(\frac{W_k^{Co}}{W_k^{Com}} - \frac{W_k^i}{W_k^{im}} - d_{min}\right)}\right)^2} \cdot \frac{\partial B}{\partial P_q} \quad k=1,..,8 \tag{16}$$

donde:

$$\frac{\partial B}{\partial P_q} = \frac{X_j^{Co} W^{Com} - X_j^{Com} W^{Co}}{(W^{Com})^2} - \frac{X_j^{i} W^{im} - X_j^{im} W^{i}}{(W^{im})^2}$$

Lo cual nos permite calcular los incrementos necesarios para cada peso Pk a partir, claro, de una aproximación inicial.

Las pruebas realizadas con las bases de conocimiento incluidas en el anexo posibilitaron el ajuste de la función de similaridad, si bien hemos detectado ciertos problemas de desbordamiento en las operaciones a realizar, y problemas de estabilidad en el método cuando la base de conocimiento no suficientemente robusta.

5 Autoconocimiento en el Sistema

En esta sección se comentarán determinados parámetros y procedimientos que nos permiten inferir el estado de funcionamiento del Sistema. Es decir, detectar posibles malfuncionamientos, carencias o lagunas en la codificación de algunas partes del dominio, inconsistencias, degradación del rendimiento, exceso o defecto de conocimiento, etc... La discusión tendrá básicamente un carácter cualitativo en lugar de cuantitativo (hablaremos de que esto es preferible sobre esto otro, o que cuanto mayor sea X mejor) a pesar de que nuestra intención era la de proporcionar una serie de medidas objetivas que facilitaran un análisis cuantitativo. La principal dificultad reside en la propia naturaleza de la Herramienta que se está desarrollando. Como tal herramienta, debe ser lo suficientemente flexible como para poder adaptarse a los distintos dominios o aplicaciones para los que está diseñada. Y la codificación del conocimiento relativo a un cierto dominio no puede evaluarse con las mismas escalas de medida que para otra codificación incluso del mismo dominio. ¿Porqué? porque en la traducción de la información desde un esquema de representación no computacional (incluimos aquí tanto al experto humano como a cualquier posible fuente de conocimiento escrita o de cualquier otro tipo) a uno computacional intervienen ciertos procesos cognitivos que están sujetos inevitablemente a las apreciaciones subjetivas de los participantes humanos. Sin embargo, consideramos que el análisis tiene un valor indudable ya que facilitará la tarea del Ingeniero del Conocimiento a la hora de evaluar el rendimiento, validar el sistema, y especialmente a centrar el o los problemas de representación que se pueden producir en el Sistema.

Incluso la utilidad del análisis puede verse desde una doble vertiente (ver figura 28). Por un lado el estudio de los valores de los parámetros del Sistema nos permite deducir el estado del mismo; y por otro el estado del Sistema debe quedar reflejado en los parámetros que lo caracterizan. Mientras que la primera acepción nos será útil para tareas de verificación y validación del Sistema, la segunda servirá para simular y predecir el efecto que podría tener la modificación, inclusión, o eliminación de algún elemento de representación (concepto, caso, relación, etc..) sobre el Sistema. Por ejemplo, podríamos, mediante el estudio de ciertos parámetros, decidir si una nueva experiencia debe ser almacenada o no en la memoria de casos.

En los próximos apartados discutiremos las cualidades que debe poseer un Sistema correctamente definido, los parámetros que utilizaremos para evaluar al Sistema, y los distintos estados en los que se puede encontrar. Finalmente profundizaremos en las relaciones que existen entre todos estos actores.

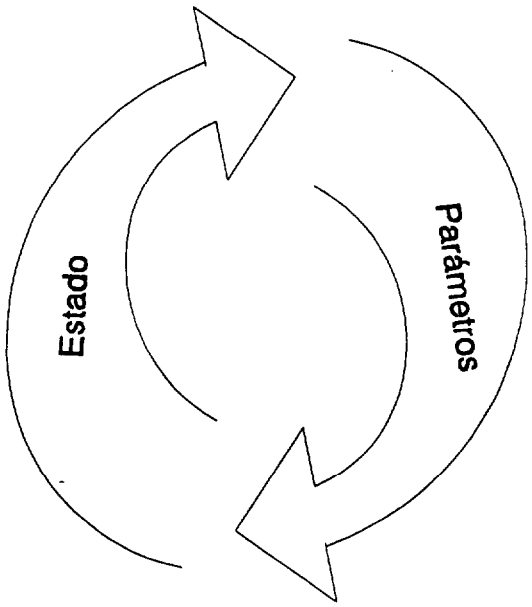


Figura 28. Realimentación entre el estado y los parámetros del Sistema.

5.1 Sistema bien definido

Decimos que un Sistema está bien definido cuando presenta las cualidades de completitud, homogeneidad (o compensación), y corrección.

Un sistema está completo cuando toda la información definida en él permite un funcionamiento eficiente del mismo. El sistema no tiene carencia alguna de información (tiene todo lo que necesita), y tampoco le sobra información (no existe unidades de información que puedan distorsionar el funcionamiento del Sistema). Esta definición no se basa en una definición exhaustiva de todo el conocimiento del Dominio (usando la teoría de conjuntos sería una declaración por extensión), sino de sólo aquella porción que permite el funcionamiento del sistema (definición por comprensión). Las estructuras que componen el sistema no representan individualmente un modelo del dominio, sino que es la conjunción de ellas la que globalmente constituyen una única representación del mismo.

La homogeneidad o compensación exige un correcto balanceo en las tres estructuras del Sistema, de forma que la aportación de cada una de ellas sea lo más constante posible. La robustez del sistema frente a carencias o ruido en la información que se suministra en relación con una estructura determinada será absorbida más fácilmente por las otras si la estructura que contiene el error no influye excesivamente sobre la función de similaridad del

sistema. Por ejemplo, si tenemos una dependencia excesiva en la red causal debido a una descompensación en los pesos que evalúan la similaridad causal, la inclusión de un sólo dato erróneo que esté implicado en alguna relación causal puede hacer que casos irrelevantes compitan por ser solución.

La información errónea puede desestabilizar cualquier sistema. En nuestro caso, la información errónea desequilibra las relaciones que existen entre las estructuras durante el esfuerzo por establecer los pesos de la función de similaridad, a la vez que el sistema es evidentemente inestable. El grado con el que el sistema se ve afectado por la inclusión de un dato erróneo (durante la fase de definición) va a depender de la relevancia de dicho dato en el dominio sobre el que se está aplicando, y por lo tanto, no es posible cuantificarlo a priori. En este sentido, se podría dar el caso de que el dato erróneo fuera totalmente irrelevante, o que influya de alguna manera en el ajuste del sistema. En cualquier caso, es poco probable que un sólo dato erróneo pueda provocar un serio descalabro del sistema.

5.2 Parámetros

La tabla 16 muestra los parámetros disponibles para evaluar la calidad del sistema. Los hemos clasificado en dos categorías en función de su origen. Tendremos por una parte los que se derivan de la función de similaridad, y por otra los derivables de las estructuras de representación.

PARÁMETROS	
Sobre la Función de Similaridad	
	Prop
	MZC
	Dif
	Violación de Relación Básica
	Descompensación en Pj
Sobre las Estructuras	
	Distancia a la media de utilización de · Relaciones causales · Conceptos · Relaciones Jerárquicas · Nodos con representante causal
	Elementos de representación más usados

Tabla: 16

$Prop_i$ representa la cantidad de información que está en el caso *i* para una combinación de entrada Ca (es una normalización porcentual del valor que devuelve la función de similaridad FS), y evidentemente nos da una idea de la similaridad entre dos situaciones. *Dif* es la distancia del caso ganador al primer caso que pertenece a una categoría de solución distinta, lo cual se puede asimilar con el grado de robustez o confianza del Sistema. *MZC* es la máxima zona en común, que nos indica el grado de solape entre dos clases de soluciones. Una violación en alguna de las relaciones básicas entre los pesos de la FS (ver tabla 15) o una descompensación entre los pesos de dicha función nos permitirá identificar determinadas carencias y deficiencias en el conocimiento representado. Las distintas medidas que se derivan de las estructuras de representación serán especialmente útiles para investigar posible acciones correctivas sobre alguno de los problemas identificados. Su utilidad es a nivel estadístico y aisladamente no aportan apenas información. Por eso las estudiaremos integradamente con el resto de las medidas y estados involucrados.

5.3 Interpretación del Estado de las Estructuras representacionales

A continuación evaluaremos los distintos estados en los que se pueden encontrar cada una de las estructuras representacionales, y discutiremos como se reflejan dichos estados en los parámetros del sistema. Por motivos de simplicidad en el estudio, cuando analicemos el estado de una estructura, supondremos que el resto de la definición del Sistema es consistente a no ser que se especifique lo contrario.

Base de Casos

Una base de casos puede encontrarse básicamente en tres situaciones distintas: sin deficiencias, no completa, no compensada, o no correcta.

El mejor indicio de que una Base de Casos no es deficiente es que el Sistema proporcione una respuesta adecuada en todas las consultas. Además, suponiendo que podamos asimilar el comportamiento de *prop* y *dif* al de una distribución normal, las desviaciones típicas deberían ser muy pequeñas, dándonos de esta forma una medida de la bondad o calidad de la Base de Casos. En general, todos los parámetros del sistema deben tener desviaciones típicas pequeñas, o, para el caso de las relaciones entre los Pj, deben ser correctas y estar compensadas (especialmente contx1 y contx2).

Una Base de Casos no está bien constituida cuando le falta algún representante, o cuando tiene un exceso de representantes para alguna clase. Este último caso no implica necesariamente una degradación excesiva del Sistema. En principio, una deficiencia de este tipo sólo genera una mayor computación y no debe ser excesivamente maligna (siempre y cuando el caso de más sea un episodio consistente). La detección de este tipo de situaciones es relativamente sencilla de realizar mediante el estudio del vector solución: la situación redundante debe estar siempre en las cercanías del representante genuino de la clase. Aunque es obvio que existen unos representantes (casos) mejores que otros para ilustrar una cierta clase de sucesos, un caso redundante no debe nunca interferir con las otras clases del dominio. En todo caso, podría producir un ligero descenso en cualquiera de los parámetros del Sistema, pero nada más. Cuando una base de casos esta incompleta, el sistema fallará tarde o temprano. Este hecho sólo es detectable mediante la prueba sistemática de la Base de Casos. Tendríamos que distinguir dos posibilidades: primero, que el sistema sea incapaz de recuperar ninguna situación. Esto nos permitiría detectar la carencia, y posiblemente incluir un nuevo representante en la Base. La segunda posibilidad es más perjudicial: a pesar de que se trata de una situación no prevista, el Sistema recupera una situación de la memoria, permitiendo que se construya una solución errónea. Esto sólo es posible si además el caso recuperado está mal definido (está incompleto o contiene información errónea). Cuanto más incompleta esté una Base, menor peso específico tendrán las dimensiones asociadas a la Base de casos (contx1, y contx2) ya que la capacidad predictiva que se le supone a estas dimensiones no podrá desarrollarse completamente.

Por compensación entenderemos que la Base venga descrita por los mejores representantes de las clases de soluciones, y que cada caso de memoria contenga sólo aquella información que fue considerada como relevante para su resolución. Un buen representante es el que permite cubrir un mayor número de situaciones posible con el mayor ahorro cognitivo. Depende del proceso de Ingeniería del Conocimiento la selección de buenos representantes. Cuanto mejor sean mejores serán los parámetros del Sistema. Por otro lado, un caso debe estar compuesto de los descriptores que fueron útiles durante su resolución. Incluir información adicional irrelevante puede producir cualquier deterioro sobre los parámetros del Sistema. Por ejemplo la *prop* media será menor en los casos descritos con información irrelevante que en los otros. El nivel de solape entre las clases también será mayor. El ajuste y homogeneidad de los Pj será más difícil de conseguir porque ahora será necesario neutralizar el efecto de los datos que sobran, etc.. Mediante el análisis de las medias de utilización de los conceptos implicados en las recuperaciones sería posible determinar si son necesarios o no para cada situación.

Difícilmente un dato erróneo en la definición de algún caso puede desestabilizar al Sistema, debido a la interconectividad de toda la información del Dominio. Esta posibilidad va a depender del grado de importancia del dato en cuestión. Cuanto más significativo es un

dato, más interrelacionado estará y mayor podrá ser el perjuicio que cause al caso que lo contiene (básicamente inhibirá algunas relaciones causales, o, en menor medida, facilitará otras). Si el resto del Sistema está bien definido, un dato erróneo muy significativo puede impedir que pueda alcanzar una solución, pero difícilmente facilitará que se de una respuesta errónea. En general, los parámetros del Sistema se degradarán ligeramente (dependiendo del número de casos de la base, y de la importancia del error). Para el caso afectado, puede que se obtengan picos negativos en *dif* (y en prop, naturalmente), pudiéndose llegar a valores de *dif* negativos cuando el dato provoque que el caso no supere su mzc.

Jerarquía de Conceptos

Los estados en los que se puede encontrar una jerarquía de conceptos son similares a los de la Base de Casos, aunque con ciertos matices en el significado de algunos estados. En general, los problemas asociados a la Jerarquía de Conceptos se refieren a la *organización* del modelo conceptual. Decidir qué es un concepto y como relacionarlo va a influir, como todo, en las puntuaciones que se podrán alcanzar con la FS.

La jerarquía contendrá aquellos conceptos que se nombren en las descripciones de los casos, y también otros conceptos abstractos que no necesariamente estarán incluidos en la definición de los casos. La organización de todos los conceptos del dominio va a facilitar la abstracción de los descriptores de entrada, para deducir nueva información relevante a la situación actual. Es por ello, que la 'calidad' de la organización de la información (en este caso los conceptos del dominio) va a influir decisivamente en la eficiencia del Sistema.

Una Jerarquía de Conceptos bien definida va a quedar reflejada en unas puntuaciones máximas mayores a la vez que un equilibrio en el resto de los parámetros del Sistema. Se maximizarán los valores de Wij para las dimensiones causales D3, D4, D5, y D6 (los valores concretos, no necesariamente los pesos asociados Pj) porque gracias a la propagación ascendente, se identificarán un mayor número de nodos causales, y por consiguiente las relaciones asociadas.

Un modelo conceptual recargado es aquel que tiene conceptos y relaciones que realmente no son necesarias para realizar las inferencias. Estas relaciones no es que sean semánticamente erróneas, sino que no son adecuadas para el o los contextos que se están contemplando. Que determinadas relaciones se activen 'fuera de contexto' puede falsear las dimensiones de la similaridad alternativa (D2 no alcanza la relevancia debida) y las asociadas con la red causal. Estas últimas pueden ser perjudiciales por cuanto su relevancia en el sistema de ponderación puede impedir que se alcancen soluciones. Como cualquier problema,

los parámetros se verán afectados en la medida en que proliferen las relaciones redundantes. Los conceptos redundantes no suponen ningún tipo de carga adicional para el Sistema excepto los de almacenamiento de su definición. Cuando el problema es que no se han definido determinados conceptos y sus relaciones asociadas, la dimensión que se verá más afectada es la de similaridad alternativa D2, la cual tampoco es una de las más significativas del modelo. Cuando la carencia es la de un concepto que puede servir de puente entre conceptos de bajo nivel y otros más abstractos y ligados con la red causal, la carencia puede ser más grave. Cuanto mayor peso tenga la relación o relaciones que no se puedan activar para una o varias situaciones mayor será el problema. En principio este problema no puede ser detectado si la falta no es muy significativa. La falta o ausencia de una relación determinada puede dejar 'aisladas' algunas zonas del modelo causal. Como consecuencia, las relaciones causales en las que intervengan dicho nodos aislados nunca podrán verificarse si estos nodos son nodos abstractos (elaborados). Este problema es fácilmente detectable mediante el análisis de las medidas de utilización de las relaciones y nodos afectados.

El modelo conceptual está descompensado cuando algunas áreas del modelo están descritas con una mayor riqueza que otras (lo que se traduce mayor similaridad conceptual alternativa, y causal directa y alternativa por las relaciones de abstracción). El efecto es un aumento de la desviación típica de las *prop* (y por consiguiente de *dif*), ya que el valor de la FS para los casos en los que interviene un cuerpo de conocimiento aislado será bastante inferior que en cualquier otro caso. Esto conduce a un mayor solape entre las clases y por consiguiente en una mayor dificultad a la hora de encontrar unos Pj adecuados. Las medidas de utilización proporcionarán los indicios suficientes para detectar estas carencias

La discusión acerca del efecto de tener relaciones entre conceptos que son erróneas (que no son no verdad) es similar al realizado para la base de casos. Puede suceder cualquier cosa dependiendo de la relevancia del enlace erróneo, aunque la probabilidad de que el error sea significativo es insignificante.

Red Causal

Al igual que con las otras estructuras, una red causal puede estar bien definida, incompleta, descompensada, o incorrecta. Sin lugar a dudas, esta es la estructura más importante del sistema; importancia que queda reflejada en el número de dimensiones de la FS que tiene asociada, y en los pesos que reciben dichas dimensiones, que para D4 será el más alto del sistema. Lo cual quiere decir que cuanto mejor esté definido el modelo causal, mejor será el rendimiento del sistema, y que cualquier deficiencia sobre su definición tendrá un efecto mayor que uno similar en otra de las estructuras.

Recargar un modelo causal con relaciones que están fuera de contexto puede desvirtuar la ponderación de las dimensiones asociadas de igual forma que se producía con los enlaces de la jerarquía de conceptos. Si la red está incompleta el rendimiento del sistema se degrada en la medida en que falten relaciones significativas (que ayudan a separar o distinguir entre clases solapadas). Como veremos posteriormente, completar el modelo causal será siempre una de las opciones preferidas a la hora de aumentar o mejorar el rendimiento del Sistema.

También la discusión realizada acerca de la descompensación y definición de relaciones erróneas de la jerarquía conceptual es válida en este entorno. Un desequilibrio en la definición del modelo causal hacia determinados aspectos o ámbitos podría impedir la identificación de clases que comparten gran cantidad de información. Además se producirá un aumento en las desviaciones típicas de los parámetros del Sistema, y se complica el proceso de ponderación de la FS. El estudio de las medidas de utilización de relaciones causales por los casos pueden ser suficientes para identificar este tipo de deficiencia.

La información causal errónea tendrá un efecto multiplicador debido a la relevancia que tiene en el cálculo de la FS, aunque tampoco deberá ser determinante para el rendimiento del Sistema.

Resumiendo, podremos decir que se degrada de distinta manera el funcionamiento del Sistema dependiendo de la deficiencia que se produzca en la representación del Dominio. En general, los parámetros se deteriorarán dependiendo del tipo y localización del defecto. Así mismo, la conjunción entre la medida y la dimensión de la FS que se deteriora nos permitirá centrar el problema, aunque todo depende de la gravedad del mismo. No será posible detectar el problema (o al menos no será fácil de detectar) si la relevancia de la información implicada es pequeña. Lejos de ser una desventaja es una ventaja que el Sistema sea capaz de absorber errores menores producidos en la codificación del conocimiento; errores que, por otro lado, seguro que se producirán durante el desarrollo de cualquier sistema basado en el conocimiento.

5.4 Interpretación de los valores de los parámetros

De la misma manera que hicimos con los estados representacionales, discutiremos en las próximas secciones acerca del significado de los valores que pueden tomar los parámetros del Sistema. Este análisis es complementario del anterior, proporcionando la perspectiva contraria, y añadiendo algunos aspectos que no podían estudiarse desde el otro punto de vista.

A no ser que se especifique lo contrario, nos referiremos siempre a valores medios.

La *mzc* es el mejor indicador del solape que existe entre los casos de la base. Por ello, cuanto menor sea su valor medio mejor, aunque con ciertas precauciones porque podría significar que el caso está descrito con información irrelevante. Si este no es el caso, valores bajos en el *mzc* facilitarían el razonamiento analógico permitiendo que se resuelva un mayor número de situaciones por cada representante en la base de casos. Por contra, cuanto mayor sea, mayor será el solape entre clases, indicando que las categorías están más cerca unas de otras.

Es fácil detectar casos problemáticos estudiando su *mzc* particular. Cuando el margen de libertad es demasiado estrecho entre dos casos de clases distintas, puede hacerse necesario llevar a cabo un proceso de desacoplamiento mediante el enriquecimiento de los descriptores de los casos, y la revisión de las estructuras asociadas a los mismos.

Prop

Como se ha venido comentando, *prop* proporciona una medida sobre la cantidad de información que se utiliza para resolver una situación. Evidentemente, sólo tiene sentido para casos de prueba previstos al efecto (con los casos de la base su valor sería del 100%). Es un buen indicador de la bondad de la representación de los casos de la base. Cuando los casos están bien elegidos, capturan la 'esencia' de la clase a la que representan, de manera que los casos de consulta deben poseer la máxima cantidad de información en común con los de memoria. Un valor medio excesivamente bajo podría ser un claro indicio de que los casos de memoria están descritos con información irrelevante que realmente no aportan nada a la solución del caso.

La robustez del Sistema también puede evaluarse comparando los valores de *prop* respecto al de los umbrales *mzc*. Valores similares son indicios de debilidad, y lo contrario de robustez del Sistema. La diferencia entre *prop* y el umbral *mzc* proporcionan el margen de funcionamiento del sistema. Es un índice de los grados de libertad que existen frente a errores en la definición de los casos y demás estructuras de representación, o disponibilidad de información durante la consulta. Por eso, conviene maximizar esta diferencia en lo posible. Cuando el margen es menor de lo esperado se deben tomar las medidas necesarias para favorecer el razonamiento analógico, buscando alguna de las deficiencias comentadas en la sección anterior (elección de un mal representante, conocimiento causal demasiado específico, insuficiente jerarquización de la base, etc..) ayudándose de alguna de las medidas disponibles que existen sobre las estructuras del conocimiento (ver tabla 16)

Un valor alto en la media de *prop* debe ir acompañado de una desviación típica pequeña. La desviación típica alta indica que el Dominio no requiere de la misma cantidad relativa de información para todas las situaciones, lo que unido a valores medios bajos de *prop* significa una descompensación en las estructuras de representación. Es decir, el valor de *prop* oscilará en función de qué descriptores se utilicen para definir un caso. Si el esquema de representación está descompensado, unos descriptores generarán una aportación muy superior que otros que son igualmente importantes semánticamente hablando.

Dif

Un valor negativo de *dif* para algún caso de memoria indica un fallo en el Sistema. Este parámetro también se puede considerar como una medida del solape entre dos situaciones, pero desde el punto de vista de una tercera situación (la de consulta), lo cual puede ser bastante distinto que para el caso del umbral. Para mantener la coherencia de la definición de esta medida, sólo deben utilizarse situaciones que den lugar a una sola solución.

Al igual que sucedía con los umbrales, valores altos son indicios de robustez en el Sistema. Además alcanzar desviaciones Típicas bajas de *dif* podría ser incompatible con valores altos con lo que habría que llegar a un compromiso si eso se produjera.

Violación de Relación Básica en FS

pcon1 > pcon2

Una violación de esta relación significaría que los casos de consulta '*prefieren*' similaridades con conceptos vecinos que con los conceptos coincidentes. Dicho de otro modo, se prefieren lo que no se tiene, y lo que se tiene no ayuda a discernir. Este problema surge cuando se han seleccionado mal los conceptos que describen la mayoría de los casos de memoria, lo cual es altamente improbable que suceda.

pcau < prela1 > prela2 >= prela3

Que algo tenga mayor importancia que prela1 vuelve a carecer de sentido semántico. O no existen suficientes rela1, o los casos representantes han sido mal elegidos. También es posible que el conocimiento causal codificado sea demasiado específico, posibilitando que sólo se verifiquen las relaciones para los casos de la base; o todo lo contrario, que las relaciones causales sean demasiado genéricas, perdiendo de esta forma su capacidad discriminatoria.

Tampoco tiene sentido que prela2 < prela3, ya que se favorece aquello que no se conoce frente a lo que si.

pcontx1 < pcontx2

La violación de esta relación es un indicio de la presencia de atributos irrelevantes. Si cuando se encuentra una coincidencia del tipo <c,a,v> falla, es que ese atributo estorba.

Descompensación en los pesos

Cuanto más diferentes son los pesos, más dependientes son los casos de esa dimensión, y más frágil es el Sistema ante una carencia de información sobre esa información. Implícitamente es un indicador de una debilidad del resto de las estructuras frente a la predominante.

Capítulo IV

Discusión de Resultados y Conclusiones

El último capítulo está dedicado a comentar las principales conclusiones a las que hemos llegado tras la realización de este trabajo. Organizamos este tema en cinco apartados con los siguientes contenidos:

1 Descripción de las principales características de la propuesta, muchas de las cuales ya se han discutido en capítulos precedentes. Esta vez también analizaremos al Sistema desde el punto de vista funcional: ¿cuales son las tareas que puede afrontar este Sistema?. Finalizamos con una descripción del prototipo actualmente desarrollado, qué capacidades están implementadas y cuales están simuladas.

2 Limitaciones de la propuesta actual. Medidas a tomar tendentes a minimizar o eliminar su incidencia sobre el Sistema.

3 Discusión de las implicaciones a las que puede dar lugar esta propuesta, en relación con diversos aspectos de los Sistemas Basados en el Conocimiento.

4 Extensiones y conclusiones del trabajo realizando, entre las que se encuentran la realización de un trabajo de expansión del prototipo, para dotarle de una funcionalidad plena, y la investigación de otras posibilidades y características adicionales.

1 Descripción Funcional y Principales Características

Hemos completado el análisis de una nueva arquitectura CBR en la que se integran diferentes formalismos representacionales, y en la que se propone una nueva función de evaluación de la similaridad. Al mismo tiempo hemos definido un proceso de resolución capaz de seguir múltiples líneas de razonamiento simultáneamente.

1.1 Esquema de Representación

El esquema de representación propuesto permite la codificación de diversos aspectos del conocimiento de una forma organizada y coherente. Cada una de las estructuras representacionales que componen el Sistema abordan una dimensión distinta del conocimiento. De esta forma es posible modelizar un dominio de trabajo mediante la combinación de diversos tipos de conocimiento: el episódico, el causal, y el conceptual. El episódico organiza la experiencia en unidades de conocimiento globales (casos), proporcionando las justificaciones genéricas implícitas en las descripciones mismas de los casos 'tipo' almacenados en memoria. Las descripciones de los casos se componen de un conjunto de objetos activos (descritos con valores concretos). Los casos de memoria también aportan los caminos que se siguieron durante su resolución, los cuales serán adaptables a las circunstancias que impone la situación actual. La modelización de un sistema mediante relaciones causa-efecto implica la descomposición en unidades funcionales en las que se codifican determinados aspectos puntuales del conocimiento. Estas unidades de información no tienen porqué ser del mismo tamaño[1], ni tampoco precisan de una relación directa con la memoria episódica. Una relación causal implica una ordenación parcial entre elementos, lo que puede también utilizarse para reconstruir caminos de resolución, o proponer nuevas vías de exploración durante la resolución. De esta manera, cuando ningún caso de memoria es capaz de aportar el camino que se siguió para su resolución, puede utilizarse la información almacenada en la estructura causal para proseguir con la construcción de una nueva solución. La última estructura representacional refleja la estructura jerárquica del modelo conceptual del dominio. Básicamente nos proporciona los mecanismos necesarios para construir las abstracciones necesarias para el proceso de resolución. Las abstracciones o elaboraciones de

[1] Por tamaño de la unidad de información nos referimos aquí al nivel de abstracción del elemento. Un elemento con un nivel alto de abstracción tiene un mayor tamaño que uno con nivel bajo, porque aquel abarca a un mayor número de unidades de información.

conceptos permiten articular mecanismos de razonamiento analógico mucho más potentes de lo que se podría conseguir únicamente con características de bajo nivel.

La independencia de estas estructuras nos ha permitido definir claramente los cometidos de cada componente, resultando en estructuras simples y sencillas de gestionar.[2] Pero esto nos obliga a articular los mecanismos de comunicación entre ellas. Nuestra propuesta centra el Sistema en el '*Concepto*' como unidad fundamental que está presente en todas las estructuras (figura 29). Estas *Unidades de Información*, tal como fueron definidas en el Capítulo II nos permiten identificar y definir cualquier objeto (tangible o intangible) perteneciente al dominio que se esté modelando. O dicho de otra manera, cada Unidad de Información se puede asociar con un concepto del Dominio. La Jerarquía de Conceptos especifica las relaciones jerárquicas o estructurales que existen entre todos los conceptos identificados en el Dominio, estén presentes o no en alguno de los casos de memoria, pertenezcan o no al modelo causal. Los conceptos también son el componente fundamental en el modelo causal junto con los enlaces que los relacionan. Y finalmente un episodio de la memoria viene descrito por un conjunto más o menos extenso de unidades de información. De esta forma hemos definido un único canal de comunicación entre las estructuras: el Concepto. Cualquier petición de información debe realizarse a través de las unidades de información. Un módulo de control se encarga de redirigir la petición a la estructura correspondiente. Estos son algunos de los requerimientos de información que se realizan a la memoria:

- ¿Cuáles son los casos relacionados con el *concepto X*? (dirigida a la Memoria Episódica)
- ¿Cuál es la zona de influencia del *concepto X*? (dirigida a la Jerarquía de Conceptos)
- ¿En qué relaciones causales interviene el *concepto X*? (dirigida a la Red Causal)

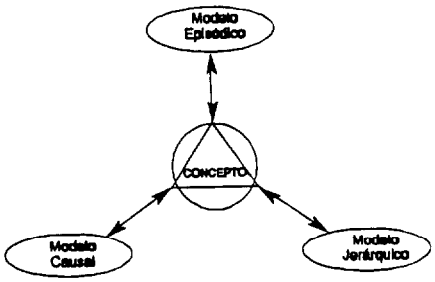


Figura 29

[2] Estas y otras ventajas de una clara separación de las estructuras representacionales ya fueron enumeradas en el primer apartado del Capítulo II.

El planteamiento de un Sistema híbrido, la independencia de sus estructuras, y el mecanismo de coordinación, características principales del esquema de representación, traen consigo una serie de ventajas como son: oportunismo en la utilización de la información, un ahorro cognitivo en la modelizacion del conocimiento, la minimización de los efectos de la información errónea, la posibilidad de trabajar con modelos incompletos y una simplificación de los esfuerzos de desarrollo.

El oportunismo implica la posibilidad de acceder a todo el conocimiento relacionado con una unidad de información tan pronto como se tiene conocimiento de ella, integrando dicho conocimiento dentro del proceso de resolución inmediatamente. A diferencia de los algoritmos tradicionales de los sistemas de producción (Forward y Backward), en los que la inclusión de nueva información durante la consulta no implica necesariamente el aprovechamiento de la misma, cuando se tiene conocimiento de un nuevo dato se exploran todas las implicaciones de ese dato sobre el proceso de resolución en curso, actualizándose las medidas de similaridad con los casos de la memoria, e incluso afectando al centro de atención de la resolución.

La posibilidad de distribuir el conocimiento adquirido en tres estructuras de representación conduce a un ahorro cognitivo en la modelización del conocimiento en comparación con los sistemas que sólo utilizan una memoria episódica. Un sólo caso de memoria puede ahora responder a un mayor número de estímulos al complementar su definición con la información que le proporcionan las otras estructuras.

Sin ser una contradicción con lo comentado anteriormente, a la vez que un ahorro congnitivo se produce necesariamente un efecto de redundancia de la información en el sistema muy difícil de eliminar. Lejos de ser una desventaja, esto posibilita la construcción de un sistema robusto frente a errores en la codificación de la información. Los efectos negativos de un dato erróneo pueden minimizarse al existir caminos alternativos que conduzcan a la hipótesis correcta.

Deja de ser imprescindible una especificación exhaustiva del conocimiento acerca del Dominio de implementación. La complementariedad de las estructuras utilizadas permiten soslayar las carencias que se producen durante la codificación de la información. Evidentemente se sigue cumpliendo la norma de 'cuanta más información mejor' aunque el sistema es capaz de trabajar con estructuras parcialmente definidas.

Finalmente, también los esfuerzos de desarrollo se verán beneficiados del esquema que hemos planteado en esta sección. Las labores de adquisición del conocimiento podrán diseñarse y llevarse a cabo de una forma más efectiva y eficiente, como comentaremos posteriormente. La codificación también será más sencilla por la garantía de independencia entre las estructuras representacionales.

1.2 Función de Similaridad

La Función de Similaridad (FS) sintetiza todo el conocimiento del Sistema respecto a una situación dada. La FS en sí no es novedosa. Tiene la forma tradicional de una suma ponderada de ciertas dimensiones (figura 30). Sin embargo, en lugar de utilizar como parámetros directamente las unidades funcionales de la situación actual, realizamos una transformación en el espacio de estados de la definición de la situación actual, al espacio formado por ocho dimensiones fijas definidas a priori. Esta transformación presenta las siguientes ventajas:

- Permite recoger y combinar información de naturaleza tan distinta como la episódica, la causal, y la conceptual.
- Sólo existe una dependencia entre las dimensiones[3] pertenecientes a la misma categoría (relacionadas con la misma estructura de representación), lo que nos permite mantener aisladas a las distintas estructuras no sólo a nivel de especificación del conocimiento sino que también durante todo el proceso de resolución de problemas.
- Al ser estas dimensiones de naturaleza heurística, la independencia facilita la sustitución de la tarea que realizan por otra que se considere más conveniente al dominio con el que se esté trabajando, sin que ello afecte al resto de las dimensiones de la FS.
- Mediante el desarrollo de las dimensiones alternativas, se permite que el Sistema sea capaz de propocionar la respuesta correcta frente a deficiencias o lagunas en el conocimiento definido dentro de una misma estructura representacional.
- Es posible ponderar la importancia de cada atributo asociado a una unidad funcional sin tener que utilizar un costoso y a la vez oscuro sistema de pesado individual. La FS recoge las distintas dimensiones de cada unidad funcional dentro del contexto en el que se encuentran. De esta forma, no se asigna a un atributo un peso específico dentro del dominio, sino que, dependiendo de las circunstancias que rodeen al atributo en cada caso de memoria se le asignará un peso u otro.
- Gracias al reducido número de dimensiones de la FS es posible desarrollar un método que ajuste automáticamente los pesos, facilitando así un funcionamiento más efectivo del Sistema, tal como se comentó en el Capítulo III.

En definitiva, la FS recoge las aportaciones que cada una de las estructuras es capaz de realizar respecto de la situación actual y genera una medida global de la similaridad entre esta y cada situación de memoria, teniendo en cuenta el contexto en el que se desarrolla el

[3] Las funciones de cada dimensión han sido claramente definidas en el capítulo II.

episodio, utilizando la información disponible y los mecanismos alternativos de deducción cuando ésta sea incompleta.

$$FS(c_i)_{Ca} = \sum_{j=1}^{8} W_j \, P_{ij}$$

Figura 30

1.3 Proceso de Resolución de Problemas

El proceso de resolución de problemas (PSP) es el encargado de interpretar los resultados de la FS, de asimilar traza derivacional de los casos recuperados, y de construir la nueva solución.

Los resultados de la FS se someten a las pruebas de recuperación (Tabla 14. Capítulo III), generando un *conjunto de recuperación* (definido en el Capítulo II). A partir de este momento, comienza un proceso monótono de razonamiento por el que se tratarán de asimilar las soluciones que proporcionan los casos del conjunto de recuperación, siempre y cuando el conjunto no esté vacío. El conjunto de recuperación está formado por aquellos casos que, conteniendo una cantidad de información mínima necesaria, garantizan un grado de convergencia hacia la situación actual y, simultánemente, de divergencia respecto al resto de los casos de memoria. La divergencia de un caso respecto al resto de casos de la memoria, se materializaría al complementar la definición de la situación actual con nuevos descriptores comunes con el caso estudiado.

La única condición que debe verificar un caso perteneciente al conjunto de recuperación para proporcionar su solución es que el conjunto de metas que codifica se verifique para la situación actual. A diferencia de la mayoría de los sistemas tradicionales, el sistema de puntuación resultante de la evaluación de la FS no genera un ganador con el que trabajar, sino un conjunto variable de trabajo. De esta forma se sustenta un PSP capaz de trabajar con varias soluciones simultáneamente. Al final el Sistema podrá proporcionar una solución que puede comprender múltiples soluciones almacenadas en memoria (ej. un paciente con varias enfermedades).

Otra de las características importantes del PSP consite en su capacidad de construir nuevas soluciones, o mejor dicho, nuevos caminos de resolución, distintos de cualquiera de los almacenados en memoria. Esto se consigue gracias a la capacidad del sistema de cambiar de centro de atención. Cuando el PSP no es capaz de verificar una meta en la situación actual siguiendo las mismas pautas que se siguieron con el caso recuperado, cambiará su atención

al caso más prometedor que verifique dicha meta con una jutificación distinta. Este oportunismo durante la resolución de problemas puede conducir a la generación de nuevas trazas derivacionales.

A diferencia de la propuesta de Carbonell, el Sistema no tendrá que reiniciar la consulta para intentar una vía cada vez que no se pueda verificar una meta, sino que el trabajo realizado durante la consulta se aprovechará para explorar una nueva alternativa. Esta posibilidad liberará al sistema de tener que realizar cálculos redundantes, a la vez que posibilitará que la reevaluación de la FS sea más precisa al tener más información disponible que al comienzo de la consulta.

El PSP se adapta a las características del esquema de representación utilizado y a la FS sobre la que se asienta. Es el mecanismo de funcionamiento del Sistema, implementando un mecanismo de razonamiento analógico que explota toda la información a su alcance oportunísticamente.

1.4 Aplicabilidad del Sistema

El rango de tareas que es capaz de resolver el sistema, tal y como está definido actualmente se circunscribe a las que tienen un comportamiento monótono. Estas son básicamente las tareas de Diagnosis e Intepretación. Esta limitación en los tipos de problemas que es capaz de afrontar viene impuesta por un lado por una autolimitación en el diseño del Sistema, y por otro lado, en un intento de desarrollar un Sistema incrementalmente. De todos es conocido que las tareas de Planificación y Diseño son de las más complejas dentro de los Sistemas Basados en el Conocimiento, y que las tareas monótonas son tradicionalmente las menos difíciles de resolver. Aunque posteriormente volveremos sobre este asunto, nuestro objetivo último es el desarrollo de un sistema capaz de afrontar con éxito tareas de diseño, lo que requiere de una serie de capacidades que analizaremos en una sección posterior.

De todos modos, no podemos decir que la tarea resuelta sea sencilla. Dentro de la Diagnosis, existen dominios con distinto grado de complejidad. Incluso la mayoría de las versiones actuales de las herramientas CBR comerciales sólo son capaces de afrontar los tipos más sencillos de diagnosis de problemas. De hecho se utilizan principalmente como 'help-desk'.

Nosotros hemos trabajado por resolver tareas de cierta complejidad, que no pueden ser codificables mediante meras tablas de asociación síntomas-enfermedad. El sistema es capaz de trabajar con problemas en los que el conocimiento o parte de la información de entrada

no es fiable debido a que es posible codificar información redundante. Aquellos problemas en los que el espacio de soluciones sea grande pueden resolverse siempre y cuando no sea necesario el Backtracking. Tareas que se ajustan a esta definición son:

- Las que se pueden evaluar descomponiéndolas en soluciones parciales, y que además, estas estén sometidas a un orden de pasos fijo (ej. Diagnosis + Tratamiento).
- Aquellas en las que no existe una secuencia fija de subproblemas, en las que se pueda abstraer el espacio de búsqueda (ej. algunos tipos sencillos de planificadores).
- En las que existe una interacción entre subproblemas y que pueden ser resueltas mediante métodos como la propagación de restricciones o el algoritmo del menos expuesto (least commintment).
- Las que necesitan mantener múltiples líneas de razonamiento simultáneamente.
- Las que no pueden codificarse con un único modelo de representación.

No es posible trabajar con tareas en las que es necesario realizar suposiciones (razonamiento plausible), ya que estas requieren de una vuelta atrás, o desarrollar cualquier tarea de diseño y planificación con una complejidad mediana o alta.

1.5 Prototipo y Pruebas

Se ha llevado a cabo la implementación de gran parte de la arquitectura presentada. Hemos utilizado tres pequeños dominios para probar su grado de efectividad (enfermedades gastrointetinales en pequeños animales - 14 casos, fase de subasta en un juego de cartas - 30 casos, concesion de créditos bancarios - 3 casos) .

A pesar de que no se ha utilizado un proceso de ingeniería riguroso en el desarrollo de esos ejemplos, consideramos que los resultados se han ajustado a las expectativas que teníamos inicialmente. Estos dominios nos han permitido realizar las pruebas y depurar los algoritmos vistos en los capítulos II y III. Sin embargo, no ha sido necesario implementar la totalidad de la arquitectura para demostrar la viabilidad de la misma. La tabla 17 resume las capacidades que posee el Sistema indicando las que están implementadas. Donde no se realiza ningún comentario se entiende que la implementación ha seguido la definición descrita en el Capítulo II..

El significado de las abreviaturas de la columna 'Prototipo' es el siguiente:

S: capacidad implementada

N: capacidad no implementada

P: capacidad parcialmente implementada o simulada

Capacidad	Prototipo	Comentarios
Representación		
· Base de Casos	S	
· Jerarquía de Conceptos		
- *Relaciones Padre-Hijo*	S	
- *Herencia*	N	La Herencia de Propiedades no es determinante, y sólo es útil a la hora de facilitar el desarrollo de aplicaciones
- *Abstracciones*	S	Construcción de la Zona de Influencia para cada nodo, y la Astracción hacia los nodos Padre con referencia Causal
- *Elaboraciones*	P	Simulado mediante la inclusión manual de un concepto con el resultado de la elaboración
· Red Causal		
- *Nodos OR*	S	
- *Nodos AND*	S	
- *Nodos TEST*	P	Simulado utilizando los nodos de las elaboraciones
Función de Similaridad	S	
· Ajuste Automático	S	Sólo en la fase de entrenamiento
Inferencia		
· Razonamiento No-monótono	N	Implica el desarrollo de un modelo de mantenimiento de la consistencia entre datos.
· Verificación de Hipótesis	S	Seguimiento y verificación del camino de resolución del caso recuperado
· Hipótesis Backward	P	Expansión en la red causal de las hipótesis encadenadas y en distinto orden del especificado en los casos de memoria. Se simula ordenando adecuadamente las hipótesis en los casos de memoria
· Combinación de Casos	S	Construcción de una solución mediante la recombinación de más de un caso, y múltiples soluciones.
· Búsquedas de Alternativas	S	Alternativas a la verificación de hipótesis, según la precedencia definida en el Capítlo II.
· Pruebas de Recuperación	N	Las definidas en la tabla 3.14

Tabla 17

1.6 Ejemplo de Recuperación

Para comprobar la potencialidad de la arquitectura que defendemos en esta memoria hemos desarrollado un prototipo con la situación que utilizamos para ilustrar el ciclo de consulta en el capítulo de introducción. Demostramos que es posible construir una nueva solución mediante la recombinación de varias soluciones de casos de memoria. Durante el escenario que planteamos, el sistema utilizará la solución del mejor caso disponible para construir su solución. Cuando no pueda verificar una hipótesis parcial utilizando un caso, buscará alternativas en los siguientes casos disponibles.

La base de casos utilizada se compone únicamente de dos casos (los mínimos necesarios para llevar a cabo la prueba). Las figuras 31, 32, y 33 contienen respectivamente los episodios utilizados para la prueba, la red causal, y la jerarquía de conceptos. Las estructuras y respuestas del programa se han editado y comentado para facilitar su lectura.

La inicialización del sistema da como resultado las siguientes mediciones en las dimensiones de la función de similaridad:

Caso	FS	D1	D2	D3	D4	D5	D6	D7	D8
c1 (aval)	305	18	0	13	6	0	0	12	0
c2 (interés trb.)	217	13	0	9	3	0	0	10	0

Con la introducción del caso de consulta se calculan las posibles elaboraciones, y se comprueban qué nodos causales se verifican para la situación actual. A continuación se evalúa la FS para cada uno de los casos recuperados y se calcula el cociente $FS(Ci)_{Ca}/ FS(Ci)_{Ci}$, siendo los resultados los siguientes:

Caso	FS	D1	D2	D3	D4	D5	D6	D7	D8
c1 (aval)	0,554	11	20	6	1	0	0	4	7
c2 (interés trb.)	0,525	8	1	4	0	0	0	2	7

Con el caso c1 recuperado se procederá a adaptar sus metas en el orden en el que fueron definidas. En primer lugar, se verifica la *estabilidad personal* utilizando el mismo argumento que para el caso recuperado, pasando esta información (meta y argumento) a engrosar la

descripción de la situación actual. Como se ha modificado la descripción de Ca, es necesario reevaluar la FS, siendo estos los valores actualizados:

Caso	FS	D1	D2	D3	D4	D5	D6	D7	D8
c1 (aval)	0,616	12	2	7	2	0	0	4	7
c2 (interés trb.)	0,612	9	1	5	1	0	0	2	7

El Caso ganador sigue siendo el mismo, así que se toma la siguiente meta por verificar (aval) y se comprueba que el argumento es aplicable a la situación actual. El proceso conduce a una revaluación de la FS con la siguiente respuesta:

Caso	FS	D1	D2	D3	D4	D5	D6	D7	D8
c1 (aval)	0,679	13	2	8	3	0	0	4	7
c2 (interés trb.)	0,612	9	1	5	1	0	0	2	7

La siguiente meta del mejor caso recuperado es la de *solvencia*, así que los argumentos utilizados para el episodio c1 se incluyen en la descripción de la situación actual Ca. La revaluación de FS resulta en:

Caso	FS	D1	D2	D3	D4	D5	D6	D7	D8
c1 (aval)	0,741	14	2	9	4	0	0	4	7
c2 (interés trb.)	0,691	19	1	6	1	1	0	2	7

La última meta por verificar (*trayectoria profesional*) con justificación causal no nula no es verificable utilizando los mismos argumentos que en el mejor caso recuperado (c1), por lo que se busca el siguiente mejor caso que contenga dicha meta y que utilice un argumento que se pueda verificar en la situación actual. Éste lo proporciona el caso c2, de forma que para construir la solución se han utilizado argumentos, o justificaciones que provienen de dos casos de memoria. La siguiente revaluación de FS no modifica la ordenación del caso ganador. Tras la inclusión de la última meta del caso ganador *(crédito)* finaliza la construcción de la nueva solución.

Identificación:
Id: c-1
Categoría: concedidos con aval

Triplas:
- Préstamo
  - cantidad: 1.750.000
  - plazo: 24 meses
  - propósito: estudios
  - estado: concedido
- Trabajo
  - Contrato: true
  - Tipo: Fijo
  - Salario: 225000 ptas.
  - Ocupación: Funcionario
  - Años trabajando: 10
- Cliente
  - edad: 45
  - Casado: true
  - Años casado: 20
  - Hijos: 1
  - Propiedades: true
  - casa propia: true
  - Gastos: 130000 ptas.
  - margen: cliente.gastos * 0.2- Aval
  - estado: verificada
- Solvencia
  - estado: verificada
- Estabilidad personal
  - estado: verificada
- Trayectoria profesional
  - estado: verificada

Metas
- Estabilidad personal, estado, Buena Situación Familiar
- Aval, estado, Balance >0 Y Balance > Margen
- Solvencia, estado, Balance > 0
- Trayectoria profesional, estado, Trabajo fijo
- Crédito, estado, null

Identificación:
Id: c-2
Categoría: Interés por su Trabajo

Triplas:
- Préstamo
  - cantidad: 4.500.000
  - plazo: 12 meses
  - propósito: viajes
  - estado: concedido
- Trabajo
  - Contrato: true
  - Tipo: Temporal
  - Salario: 300.000 ptas.
  - Ocupación: Otro
  - Años trabajando: 4
  - Promoción: true
  - Interés: true
- Cliente
  - edad: 25
  - buena situación familiar
  - Gastos: 150.000 ptas.
  - margen: cliente.gastos * 0.2- Aval
  - estado: verificada
- Solvencia
  - estado: verificada
- Estabilidad personal
  - estado: verificada
- Trayectoria profesional
  - estado: verificada

Metas
- Estabilidad personal, estado, Años casado > 4 Y Hijos
- Solvencia, estado, Balance > 0
- Trayectoria profesional, estado,Años trabajando <8 Y Trabajo temporal Y Interés por su trabajo
- Crédito, estado, null

Identificación:
Id: c-x
Categoría: Caso de Consulta

Triplas:
- Préstamo
  - cantidad: 2.500.000
  - plazo: 24 meses
  - propósito: otros
- Trabajo
  - Contrato: true
  - Tipo: Temporal
  - Salario: 300.000 ptas.
  - Ocupación: Otro
  - Años trabajando: 7
  - Promoción: true
  - Interés: true
- Cliente
  - edad: 29
  - Hijos: 2
  - Casado: true
  - Años Casado: 8
  - Gastos: 150.000 ptas.
  - margen: cliente.gastos * 0.2- Aval

Figura 31

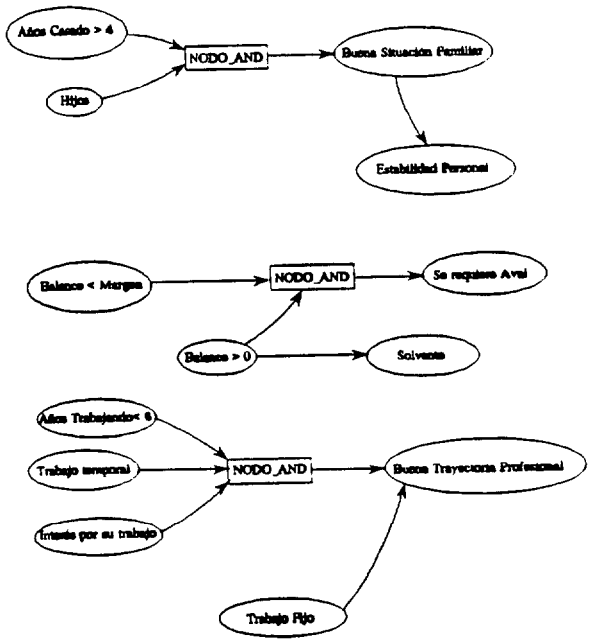


Figura 32

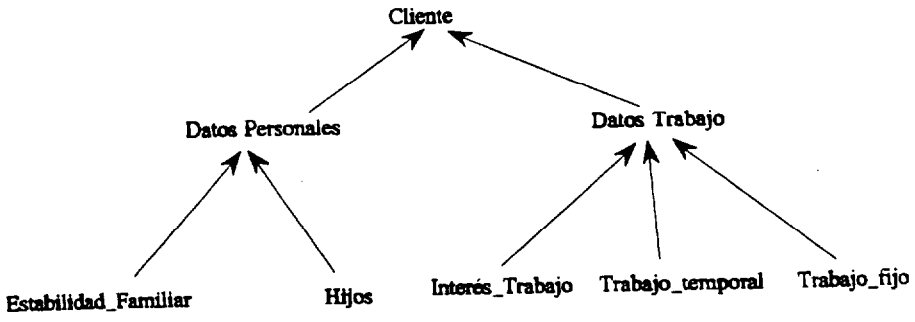


Figura 33

2 Limitaciones de la Propuesta

Antes comentamos la aplicabilidad limitada de la arquitectura a determinados tipos de problemas como los de planificación, y apuntábamos como principal causa la imposibilidad que tiene la propuesta actual de realizar un backtracking. O sea, de deshacer o retractarse de decisiones que ya han sido tomadas. Esta es la principal característica que poseen las aplicaciones de diseño y/o planificación. Estas aplicaciones se caracterizan porque se definen en base a una serie de restricciones que se deben verificar, y por el gran número de variables que deben configurarse, lo que suele dar lugar a una explosión combinatoria. Usualmente estas tareas se resuelven con algoritmos específicos que podan el árbol de estados lo antes posible, dejando sólo algunas combinaciones a comprobar sistemáticamente. El proceso de poda se suele apoyar en técnicas de vuelta atrás, la plausibilidad, múltiples líneas de razonamiento, descomposición en subproblemas, etc... El mayor coste de estos algoritmos reside en los mecanismos de mantenimiento de las dependencias que existen entre distintos datos de la consulta. De ahí surgen los TMS (Truth Maintenance Systems) que se encargan de registrar las dependencias que existen entre las variables del sistema, y de mantener la consistencia entre ellas. Así, cuando falla una suposición, o cualquier dato durante la consulta, se puede saber exactamente qué parte del razonamiento hay que reconstruir y qué puede permanecer.

Para que nuestro sistema pueda tener este comportamiento no-monótono será necesario incluir alguna técnica de TMS, y la posibilidad de ejecutar acciones en los nodos consecuentes de la red causal. Entonces, tendremos que definir un lenguaje de 'acciones' que puedan modificar datos, o inferir nueva información durante la consulta. El TMS actuaría como un demonio que registraría dependencias cada vez que se infiere nueva información (utilizando como apoyo la propia estructura causal) y que comprobaría la consistencia con las dependencias actualmente almacenadas. Cada vez que se modifica una variable incluida en la lista de dependencias sería necesario reevaluar todas las que dependan de ella.

Otra limitación del Sistema y que necesitará de un estudio más profundo es el coste computacional en la reevaluación de la función de similaridad y en el cálculo de la pruebas de recuperación. Además de que este coste es en sí muy elevado, como vimos en el capítulo III, crece al menos linealmente con el tamaño de la base de conocimiento. Nuestros esfuerzos deben dirigirse a desplazar este coste del tiempo de consulta al de compilación o inicialización manteniendo estructuras con la mayor cantidad de cálculos parciales posible.

Finalmente, las pruebas realizadas muestran que, en ocasiones, se depende excesivamente del conocimiento causal codificado, especialmente en las primeras fases del desarrollo. Aunque esto no puede ser definido como una limitación propiamente dicha, la tendencia a sobreutilizar conocimiento causal (de gran peso específico para la FS) como medio de discriminación entre clases, puede hacer disminuir la importancia relativa de las otras estructuras, haciendo al sistema menos robusto frente a errores, ruido, imprecisiones, etc.. Algo similar ocurrirá en aquellos dominios en los que una estructura no puede modelizarse de manera homogénea (cubriendo todo el espacio de estados/clases con igual intensidad). Estos 'vacíos de información' dan lugar a un ajuste débil de la FS, disminuyendo el margen de error y otra vez afectando a la robustez del Sistema. Será misión del Ingeniero del Conocimiento que desarrolla la aplicación diseñar una Adquisición del Conocimiento que facilite el desarrollo homogéneo tanto interno como externo de las estructuras representacionales. Pero sobre este asunto discutiremos algo en la siguiente sección.

3 Implicaciones

Analizamos aquí la medida en que se ve afectado el proceso de Ingeniería del Conocimiento al utilizar la arquitectura que aquí se presenta. También realizaremos algún comentario acerca de las posibilidades de aprovechar el trabajo realizado en distintos campos de la Inteligencia Artificial. Y finalmente, propondremos un modelo de autoconstrucción de la Base de Conocimiento que se apoya en esta arquitectura, el cual puede constituir por sí mismo, una vía de expansión de la investigación en CBR.

3.1 Desarrollo de Sistemas Basados en el Conocimiento (SBC)

Como cualquier software, los Sistemas Expertos se construyen siguiendo una metodología de desarrollo, que por las características intrínsecas de estos sistemas, es distinta de las metodologías utilizadas para el desarrollo del software convencional. Metodologías existen muchas, aunque nosotros, por razones obvias, nos decantamos por la que se presenta en [ROD96]. Cualquier herramienta de construcción de SBC debe ser capaz de adaptarse a las características del proceso de desarrollo del software. No tendría sentido diseñar una herramienta que exija una exesiva adaptación de la metodología de desarrollo, entre otros motivos, porque la elección de la herramienta no se debe realizar en los primeros estados del desarrollo, sino cuando existe un conocimiento suficientemente profundo sobre el dominio que permita la realización de una selección objetiva por parte del equipo de desarrollo. Por otro lado, hay que reconocer que una vez seleccionada la herramienta conviene adaptar ligeramente el proceso de desarrollo para poder explotar las características de la misma [BOO94]. Nosotros vamos a comentar como la herramienta que hemos diseñado se adapta a la metodología de desarrollo de SBC, y qué aspectos hay que potenciar del proceso de desarrollo para que la herramienta trabaje eficientemente.

La figura 34 muestra el microciclo de la metodología de desarrollo de SBC. Consiste en un proceso incremental por el que se va desarrollando el SBC mediante una serie de iteraciones. En cada ciclo se incluyen un conjunto de situaciones nuevas, de forma que al final de cada ciclo el sistema será capaz de trabajar con todas las situaciones estudiadas hasta ese momento.

Para poder llevar a cabo la primera fase, es necesario categorizar el dominio y seleccionar unos buenos representantes de las categorías seleccionadas. Para nuestra orientación, podríamos asimilar tal proceso con la selección de un conjunto de episodios a incluir en el

prototipo.

Posteriormente sigue la Conceptualización y el Análisis de las situaciones seleccionadas en la fase anterior. Según la metodología, los conceptos adquiridos deben estar perfectamente identificados, estar contrastados, y ser predictivos. Con la primera cualidad se persigue que cada uno de los conceptos del dominio esté compuesto por un conjunto de atributos y relaciones funcionalmente útiles; con la segunda, que no existan ambigüedades en las definiciones; y la tercera potencia la existencia de conceptos que podrían ser de utilidad, así como su situación dentro del modelo de manera que sean accesibles a través de sus relaciones con otros conceptos del dominio. Está claro, desde nuestro punto de vista, que el Sistema expuesto puede perfectamente modelar una conceptualización de esas características. Gracias al esquema de definición de las unidades funcionales, los conceptos pueden definirse y contrastarse. Además, la inclusión de un concepto en la jerarquía no está condicionada por su aparición previa en alguna de las otras estructuras, sino todo lo contrario. Así, es posible definir conceptos que no aparezcan necesariamente en ninguno de los casos definidos o incluso en el modelo causal. Sus relaciones dentro del modelo jerárquico facilitan la integración con el resto de los conceptos del Dominio, previendo una aparición futura en alguno de los casos de consulta.

El análisis se estructura en dos estratos, el primero compuesto por un conjunto de niveles en los que se desarrolla un esquema tradicional de análisis. Y en el segundo estrato se realiza básicamente una especificación lógica de procesos. Las herramientas de desarrollo de SBC han codificado las estructuras de tareas de múltiples formas. Por ejemplo, Nexpert Object implementa 'islas de conocimiento', ART-IM 'conjuntos de reglas'... Sin embargo, en los sistemas CBR es difícil codificar una estructura en tareas, así como las relaciones que existen entre ellas. El motivo es que, en principio, ya no es necesaria una descomposición detallada de las tareas que componen el Dominio. Pero esto no es así cuando la tarea que se quiere implementar es suficientemente compleja, o cuando hay que imponer una determinada ordenación en la ejecución de determinadas medidas. Nosotros nos valemos de las hipótesis parciales y de la red causal para dar una estructura de tareas al dominio. Las hipótesis parciales que se codifican en los casos de memoria se pueden considerar como las salidas de las tareas. La ordenación de las hipótesis dentro de un caso especifican las relaciones entre dichas tareas. Y el conocimiento causal relacionado con la hipótesis representa el cuerpo de la tarea.

La última fase, previa a la implementación es la extracción del conocimiento experto, o expertise. La heurística que se extrae en esta fase puede quedar reflejada en la especificación de nuevas relaciones (causales o jerárquicas) o en la inclusión de simplificaciones, o generalizaciones de casos en la base de casos.

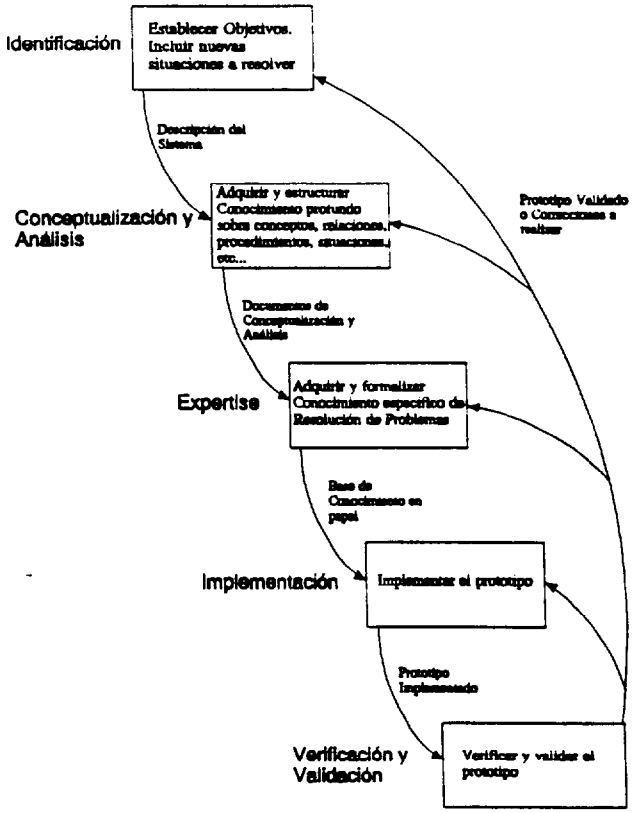


Figura 34

Cuando se sabe que se va a utilizar en la implementación de un SBC una herramienta CBR, es necesario adaptar ligeramente la metodología de desarrollo para que la capacidad analógica de la herramienta pueda ser explotada convenientemente. Durante la planificación de las sesiones de Adquisición del Conocimiento, deben predominar técnicas que traza de procesos, especialmente la generalización de situaciones y predicciones [MCG89]. Con estas técnicas el Experto y el Ingeniero del Conocimiento podrán construir buenos casos representantes de las categorías de problemas con las que se estén trabajando. Se debe potenciar la inclusión de hipótesis parciales para poder discernir entre casos muy similares y facilitar el distanciamiento entre clases de problemas. Deben excluirse de la descripción del caso aquellas características que no sean relevantes, para evitar falsear el cómputo de la FS. En cuanto a la jerarquía de conceptos, esta debe reflejar lo más fielmente posible el resultado de la conceptualización, incluso con conceptos y relaciones que, en principio, no aparezcan en ningún caso de la base. Las relaciones causales deben ser los más generales posible para favorecer el razonamiento analógico. Dado su peso específico en la FS, el modelo causal debe crecer homogéneamente. Es decir, las relaciones deben cubrir con igual intensidad todo el dominio en estudio, ya que, de otra forma, se podrían dar algunos de los efectos negativos comentados en una sección anterior y en el Capítulo III.

3.2 Reusabilidad de Técnicas en IA

Además de las técnicas ya comentadas, la modularidad y la sencillez de las estructuras utilizadas permiten la reutilización de técnicas ya desarrolladas en otros ámbitos de la IA para su aplicación a este diseño.

Las tareas de verificación pueden aplicar procedimientos como el ya comentado en el Capítulo I del CHECK, comprobando la consistencia y completitud de la red causal. Además el lenguaje de descripción de unidades funcionales es lo suficientemente restrictivo como para detectar valores ilegales en la definición de los casos, en las elaboraciones de la Jerarquía, o en los nodos TEST de la red causal. La estructura de grafo de la red causal permite utilizar técnicas bien conocidas para detectar ciclos, caminos más cortos, etc..

Por otro lado, también se pueden aplicar técnicas de organización del conocimiento conceptual (estructuración de la jerarquía de conceptos), de inducción, etc.. para ir mejorando la organización del modelo conceptual, la derivación de nuevas relaciones causales y la generalización de las existentes. También hemos visto como se pueden aplicar procesos ya conocidos para el ajuste de la Función de Similaridad.

En un futuro próximo incluiremos mecanismos de TMS para el registro y gestión de las dependencias entre datos.

3.3 AutoConocimiento

En el Capítulo II comentamos el comportamiento de varias medidas que se pueden tomar del Sistema (prop, y dif especialmente) y como este comportamiento quedaba reflejado en los distintos componentes. Esta puede ser la base para el desarrollo de un sistema de Gestión del AutoConocimiento (o Metaconocimiento) -SGA . A partir del valor de ciertas medidas, y en función del efecto conocido que producen sobre el sistema, es posible el desarrollo de un módulo de autoevaluación y de apoyo al ingeniero del conocimiento. Le proporcionaría información sobre la localización de inconsistencias, errores, lagunas en el conocimiento, etc.. Además podría proporcionar recomendaciones sobre como reparar las deficiencias (eliminando o generalizando relaciones, enriqueciendo las descripciones de los casos, etc...). El boceto de la figura 35 propone una posible descomposición del análisis, y muestra las recomendaciones básicas que realizaría el SGA.

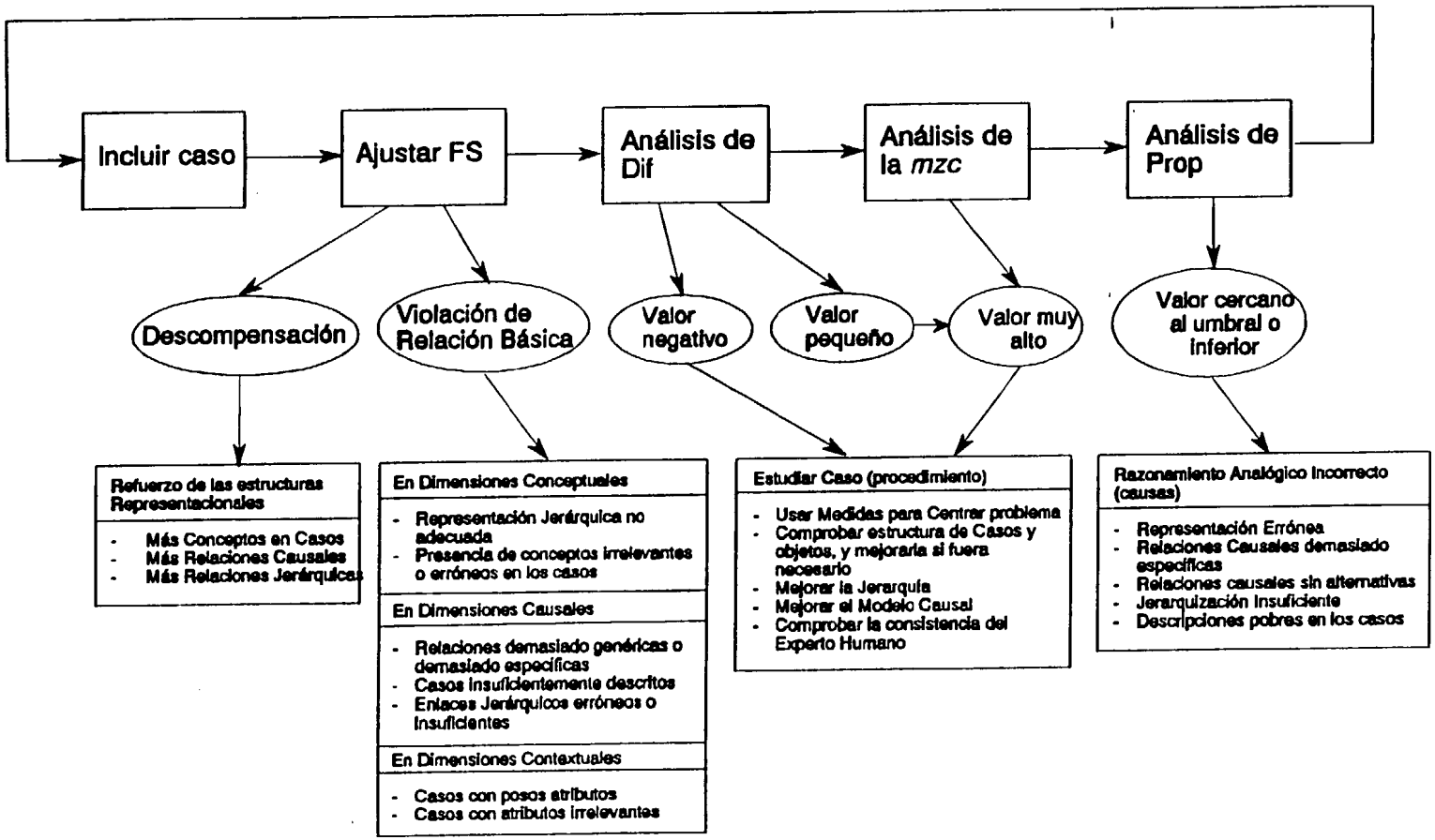


Figura 35

Después de incluir un nuevo caso en la base, es necesario reajustar los pesos de la Función de Similaridad para que el Sistema pueda asimilar la nueva situación. Durante este proceso se pueden dar tres situaciones: que el ajuste sea correcto, con lo que se pasaría a la fase siguiente; que se produzca una descompensación significativa en los pesos de la FS, con lo que las estructuras que han perdido peso específico tendrían que ser reforzadas; o que se produjera una violación de alguna de las relaciones básicas incluidas en el Capítulo III. En este último supuesto, habría que tomar las medidas pertinentes en función de la violación que se detecte, tal como se refleja en la figura 35.

Podría suceder que el mejor ajuste encontrado impida un funcionamiento correcto en todos los casos de la base. Esto podría fácilmente detectarse analizando el valor de 'dif'. Este será negativo para aquellos casos que no se han podido ajustar. Cuando esto sucede, la medida a tomar sería realizar un estudio del caso o de los casos conflictivos según el procedimiento que se muestra en la tabla. Se podrían utilizar medidas como la utilización media de conceptos, relaciones, etc. para centrar la carencia o defecto que posee el caso en cuestión. A partir de ahí habría que mejorar la estructura del caso, o de la Jerarquía, o incluso de la red causal. Si ninguna de estas medidas consigue recuperar el error, habría que cuestionarse la consistencia o validez del conocimiento extraído directamente del Experto Humano.

Posteriormente hay que realizar un análisis del grado de solape entre las clases de la base. Tal como comentamos en el Capítulo III, un solape alto se corresponde con la debilidad del Sistema. Al igual que en el caso anterior, sería necesario realizar un estudio detallado de las clases involucradas.

Finalmente, se comprueba el funcionamiento del Sistema con una batería de Casos de Prueba para poder analizar la robustez del Sistema con casos distintos de los incluidos en la Base. Cuanto mas cercano esté *prop* a la *mzc*, o cuando sea menor, será necesario replantearse si el representante de la clase está bien elegido, si está correctamente codificado, o si sería necesaria la inclusión de un nuevo representante. También podría ser conveniente el análisis de la utilización de las dimensiones causales por parte de los casos conflictivos. Formas de potenciar dicha utilización incluyen la inclusión de nuevas alternativas a las metas relacionadas con los casos problemáticos, o la adición de relaciones jerárquicas que permitan direccionar mediante las elaboraciones relaciones causales de nivel abstracto.

Una vez el nuevo caso haya superado todos estos análisis, el Sistema se encontraría en un estado estable de funcionamiento, y listo para la inclusión de nuevas situaciones si se considerara necesario.

4 Trabajo Futuro y Conclusiones

CBR es quizás el primero de los frutos de primavera que siguen al invierno en el campo de la Inteligencia Artificial, y por eso la comunidad investigadora en IA tiene buenas razones para sentirse optimista acerca de su futuro.[WAT94]

Hammond (citado en [WAT94]) divide la investigación en CBR en tres categorías en función de los objetivos que persiguen:

True Faith:	refiriendose a los temas relacionados con la psicología congnitiva y técnicas formales de IA.
Hard Core:	son los que comprueban las teorías con aplicaciones prácticas.
CBR-Lite:	herramientas comerciales super-específicas como los help-desk.

Pero, aceptando esta división, debemos mantener un flujo entre categorías para cerrar el ciclo metodológico en este campo de Investigación + Desarrollo. Es decir, de una parte, no es posible desarrollar nuevas arquitecturas sin fundamentar bien los principios de funcionamiento, y, de otra, es necesario potenciar el desarrollo de aplicaciones comerciales basadas en herramientas especializadas en tipos de tareas características, buscando fundamentalmente la corroboración de las teorías a dichas aplicaciones prácticas. Estos puntos han sido comentados y analizados como una de las auténticas carencias de gran parte de las arquitecturas CBR propuestas hasta la fecha.

Por tanto, la investigación en este campo debe expandir sus trabajos hacia más tipos de tareas y más dominios, proporcionando técnicas más potentes, especialmente en aquellos aspectos que hoy por hoy no están muy desarrollados como los procesos de adaptación y aprendizaje. Sin olvidar la optimización de las técnicas existentes para que puedan ser trasladadas a las herramientas comerciales. También debería tenderse hacia una integración con algunos de los paradigmas existentes y, que, debido a su efectividad contrastada, proporcionan garantías en su utilidad. Finalmente, no hay que olvidar el desarrollo de técnicas eficientes de validación y verificación, así como un estudio más profundo de las implicaciones que tienen este tipo de sistemas sobre el desarrollo de SBC. Y todo esto orientado a que los desarrolladores de SBC no vean defraudado su interés inicial [KOL93b].

Del análisis realizado a lo largo de todo el documento, podemos afirmar que el trabajo realizado constituye un punto de partida con múltiples posibilidades de expansión. Hemos establecido las bases de una línea de investigación que puede dar lugar a un gran número de trabajos de investigación a muy corto plazo. Nuestros objetivos de trabajo pasan primero por

completar la implementación del prototipo, haciendo un Sistema totalmente funcional, para después estar en condiciones de profundizar en los siguientes temas:

- Ampliar la utilidad del Sistema hacia tareas y dominios más complejos como son los de planificación y diseño. Para ello tendríamos que incluir alguna técnica que posibilite el razonamiento no-monótono dentro de la especificación funcional del Sistema.
- Investigar las posibilidades de la FS con la forma actual, así como con otras formas o variaciones de las ya conocidas e incluidas en otros trabajos relacionados. El objetivo en este punto se centraría en la especificación formal de las propiedades de la FS, el ajuste automático, y el desarrollo de algoritmos eficientes de evaluación.
- Completar las estrategias de adaptación independientes del dominio con nuevas reglas de adaptación dependientes del dominio.
- Afrontar el tema de aprendizaje en el sistema aplicando técnicas formales y conocidas en el campo de la IA. Asimilar nuevas experiencias reorganizando automáticamente el conocimiento en la aplicación. Es decir, decidir si eliminar o modificar alguna de las experiencias almacenadas, derivar nuevas relaciones jerárquicas y causales automáticamente, reorganizar y generar automáticamente las abstracciones de datos, etc...
- Profundizar en el esquema de Autoconocimiento.
- Estudiar y desarrollar técnicas específicas de adquisición del conocimiento, de formalización, de validación y de verificación adecuadas a estos Sistemas para que recojan las causística diferenciadora de resolución de problemas basada en el pensamiento analógico.

Por último, sólo nos queda por exponer las principales conclusiones que podemos extraer del trabajo realizado:

1. Tras una revisión profunda y crítica de los trabajos más representativos, se han encontrado las siguientes limitaciones en los mismos: a) una deficiente fundamentación cognitiva, lo que ha conducido al desarrollo de sistemas artificiosamente complicados y limitados en su capacidad de funcionamiento; b) la imposibilidad de especificar criterios de evaluación del rendimiento objetivos de las diferentes arquitecturas, consecuencia de la carencia anterior y de una excesiva particularización de las distintas aproximaciones; y c) el desarrollo de propuestas demasiado complejas como para que puedan ser integradas con las metodologías de desarrollo de SBC actualmente vigentes, y/o una falta de evolución de técnicas y métodos de adquisición y formalización del conocimiento que exploten las peculiaridades de las diversas arquitecturas.

2. Se propone un enfoque del razonamiento analógico intradominio que responde eficazmente a las carencias apuntadas en la conclusión anterior basado en el modelo cognitivo desarrollado por Gaines.

3. Se ha propuesto una nueva arquitectura funcional a partir de la cual se desarrolla una herramienta genérica aplicable sobre múltiples dominios y tareas, que, de una parte, proporciona medidas de bondad acerca del rendimiento y calidad de la formalización, y de otra, permite la integración con las metodologías tradicionales de desarrollo de SBC.

4. Se introduce un esquema de representación compuesto de: a) una memoria de casos en la que se estructura el conocimiento episódico, la cual, como estructura de mayor nivel, gobierna el funcionamiento del sistema proporcionando la unidad de recuperación (caso) y la guía de resolución para una nueva situación. b) una memoria causal que permite construir soluciones en base a las trazas derivacionales de los casos almacenados, a la vez que proporciona una justificación del razonamiento seguido durante la resolución, y aumenta el abanico de situaciones que el Sistema es capaz de resolver. Y c) un modelo conceptual que organiza el conocimiento jerárquicamente, que es utilizado como vehículo de construcción de abstraccciones y elaboraciones de la información de bajo nivel, y como nexo que articula la comunicación entre las tres estructuras.

5. Se ha definido una función de similaridad heurística en base a la ponderación de una serie de índices más que sobre pesos asignados a las características, la cual facilita el proceso de recuperación, permitiendo la definición de una serie de medidas para la evaluación del estado en el que se encuentra la recuperación y la validación del caso o los casos que se proponen para su recuperación. Desviar el centro de atención desde el *atributo* hacia el *índice* es esencial si queremos aprovechar dinámicamente la información que proporciona cada una de las estructuras representacionales disponibles.

6. La FS expande el espacio de estado formado por cada unidad de información trabajando con ocho dimensiones. Estas proporcionan mecanismos para la evaluación de la similaridad incluso cuando no existe una correspondencia directa entre el caso de memoria y la situación actual.

7 La FS evalúa en qué media está cada uno de los casos de memoria dentro de la situación actual, y no la inversa; en que medida está la situación actual en cada caso de memoria. Es decir, en el esquema de recuperación propuesto, un caso proporciona su solución en cuanto los datos que definen la situación actual se ajusten suficientemente a su descripción y sus metas puedan ser verificadas en la situación actual .

8. Se han desarrollado una serie de *pruebas de recuperación cualitativas* que optimizan el ratio cognitivo del proceso. Estas pruebas ponen de manifiesto que la recuperación de un caso se hace con la menor cantidad de información necesaria, siendo esta información aquella que garantiza la divergencia del caso recuperado respecto al resto de episodios de la base.

9. El Sistema es capaz de realizar recuperaciones contextuales y múltiples gracias al planteamiento del proceso de recuperación y de la función de similaridad propuesto.

10. El planteamiento actual del Sistema básicamente permite la resolución de tareas complejas de diagnosis e interpretación, o todas aquellas que posean un comportamiento monótono, aunque la arquitectura está abierta a modificaciones no estructurales que permitan dotarla de un comportamiento no-monótono.

11. Se ha desarrollado un prototipo CBR como herramienta utilizada para comprobar la conveniencia de las decisiones de diseño y la viabilidad global de la arquitectura propuesta.

Capítulo V

Referencias Bibliográficas

[AAM94] Aamodt, A.; Plaza, E.: Case-Based Reasoning: foundational issues, methodological variations, and system approaches. Artificial Intelligence Communications, Vol 7, nº1, 1994.

[ABE90] Abellanas, M.; Lodares, D.: Matemática Discreta. ed. RA-MA, 1990.

[ALTE89] Alterman, R.: A concept space for reasoning about cases involving event structure. Case-Based Reasoning Workshop. Morgan Kaufmann Publishers, 1989.

[ALTH95] Althoff, K; Auriol, E.; Barletta, R.; Manago, M.: A review of industrial Case-Based Reasoning tools. AI Intelligence. Gran Bretaña, 1995

[ASH86] Ashley, K.: Compare and Contrast, A Test of Expertise. American Association for Artificial Intelligence, 1986.

[ASH89] Ashley, K.: Indexing and analytic models. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[BARE89a] Bareiss, R.; King, J.: Similarity assessment in a Case-Based Reasoning. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[BARE89b] Bareiss, R.: The Experimental evaluation of a Case-Based learning appretice. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[BARL88] Barletta, R.; Mark, W.: Explanation-based indexing of cases. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1988.

[BIR88] Birnbaum, L.- Collins, G.: The transfer of experience across planning domains through the acquisition of abstract strategies. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1988.

[BIR89] Birnbaum, L.- Colling, G.: Remindings and engineering design themes: a case study in indexing vocabulary. Case-Based Reasoning Workshop. Morgan

Kaufmann, Publishers, 1989

[BLA91] Blau, L.- Bonissone, P.- Ayub, S.: Planning with dynamic cases. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1991.

[BLU94] Blumenthal, B.; Porter, B.: Analysis and empirical strudies of derivational analogy. Artificial Intelligence 67, 287-327, 1994.

[BOO94] Booch, G.: Object-oriented analysis and design with applications. The Benjamin Cummings Publishing Company. 2ª edición, 1994.

[BRAD88] Bradtke, S.: Some experiments with case-based search. Case-Based Reasoning Workshop. Morgan Kaufmann Publishers, 1988.

[BRAN91] Branting, K.: Exploiting the complementarity of rules and presedents with Reciprocity and Fairness. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1991.

[BUR89] Burke, R.: Understanding and responding in conversation case retrieval with natural languaje. Case-Based Reasoning Workshop. Morgan Kaufmann Publishers, 1989.

[CAL91] Callan, J.- Fawcett, T.- Rissland, E.: Adaptive Case-based Reasoning. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1991.

[CAR86] Carbonell, J.: Derivational Analogy: A Theory of Reconstructivo Problem Solving and Expertise Acquisition. Machine Learning, An Artificial Intelligence Approach, vol. 2, Morgan Kaufmann, 1986.

[CAS95] CASL: Creating a Case-Based using CASL. UW Aberystwyth, February 95.

[CBR91] CBR Express 1.1. Reference Manual. 1991. Inference Corp.

[COH89] Cohen, P.: Evaluation and Case-Based Reasoning. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[COL89] Collins, G.: Plan Adaptation: a transformational approach. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[CON89] Converse, T.; Hammond, K.; Marks, M.: Learning modification rules from expectation failure. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann,

Publishers, 1989.

[DAY90] Dayhoff, J.: Neural Network Architectures. An introduction.Van Nostrand Reinhold, 1990.

[DON89] Donahue, D.: OGRE: generic reaoning from experience. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[ELG93] El-Gamal, S,- Rafeh, M.- Eissa, I.: Case-based reasoning algorithms applied in a medical acquisition tool. Medical Information, vol 18. nº 2. pp 149-162, 1993.

[FOX94a] Fox, S.- Leake, D.: Using introspective reasoning to guide index refinement in Case-Based reasoning. Proc. of the sixtenth annual conference of the cognitive science society, 1994.

[FOX94b] Fox, S.; Leake, D.: Introspective reasoning in a Case-Based Planner. Proc. of the 12th national conference on Artificial Intelligence, 1994.

[FRA94] Francis, G; Ram, A.: A comparative utility analysis of Case-Based Reasoning and Control-Rule Learning Systems. AAAI-94 Workshop on Case-Based Reasoning, 1994.

[FUC95] Fuchs, B.; Mille, A.; Chiron, B.: Operator Decision Aiding by Adaptation of Supervision Strategies. En Case-Based Reasoning Research and Development, First International Conference. ICCBR-95, Lecture Notes in Artificial Intelligence, Springer Verlag, 1995.

[GAI89] Gaines, B.: Knowledge Acquisition for knowledge-based systems. Ed. Academic Press, 1989.

[GOE89] Goel, A.- Chandrasekaran, B.: Use of device models in adaptation of design cases. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[GOL89] Golding, A.; Rosenbloom, P.: Combining Analytical and similarity based CBR. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[GOL91] Golding, A.; Rosenbloon, P.: Improving Rule-Based Systems through Case-Based Reasoning, 1991

[GOO89] Goodman, M.: CBR in battle planning. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[GRI89] Grimaldi, R.: Matemáticas discreta y combinatoria. Addison-Wesley Iberoamericana, 1989.

[GUP91] Gupta, U.: Validating and Verifying Knowledge- Based Systems IEEE computer society press, 1991.

[HAI95a] Haigh, K.; Veloso, M.: Route planning by analogy. En Case-Based Reasoning Research and Development, First International Conference. ICCBR-95, Lecture Notes in Artificial Intelligence, Springer Verlag, 1995.

[HAI95b] Haigh, K.; Richard, J.: Geometric similarity metrics for case-based reasoning. En Case-Based Reasoning Research and Development, First International Conference. ICCBR-95, Lecture Notes in Artificial Intelligence, Springer Verlag, 1995.

[HAM88] Hammond, K.- Hurwitz, N.: Extracting diagnostic features from explanations. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1988.

[HAM89] Hammond, K.: Case-Based Planning. Perspectives in Artificial Intelligence. Academic Press, 1989.

[HAM91] Hammond, K.: A functional perspective on Reminding. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1991.

[HIN89] Hinrichs, T.: Strategies for adaptation and recovery in a design problem solver. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[HIN91] Hinrichs, T.- Kolodner, J: The roles of adaptation in Case-Based design. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1991.

[HOF95] Hoffman, R.: Monster analogies. American Association for Artificial Intelligence. Fal, 19951.

[HUN89] Hunter, L.: Finding paradigm cases or when is a case worth remembering?. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[JUR94] Jurisica, I.: How to retrieve relevant information?. Proc of the AAAI Fall Symposium series on relevance. Russell Greiner (Ed.), 1994

[KAM89] Kambhampati, S.: Representational requirements for plan reuse. Case-Based Reasoning Workshop. Morgan Kaufmann Publishers, 1989.

[KAS88] Kass, A.: Case-Based Reasoning Applied to Constructing Explanations. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1988.

[KAS89] Kass, A.: Strategies for adapting explanations. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[KOL87] Kolodner, J.: Extending Problem Solver Capabilities Through Case-Based Inference. Proc. 4th International Machine Learning Workshop. Morgan Kaufmann, Publishers, 1987.

[KOL88] Kolodner, J.: Retrieving Events from a Case Memory: A Parallel Implementation. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1988

[KOL89] Kolodner, J.: Judging which is the best case for a case-based reasoner. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989

[KOL92] Kolodner, J.: An Introducction to Case Based Reasoning. Artificial Intelligence Review 6, 1992.

[KOL93a] Kolodner, J.: Introduction. Machine Learning 10, 1993.

[KOL93b] Kolodner, J.: Case-Based Reasoning. Morgan Kaufmann Publishers. California. EE.UU, 1993.

[KOP88] Kopeikina, L.- Brandau, R.- Lemmon, A.: Case based reasoning for continuous control. Case-Based Reasoning Workshop. Morgan Kaufmann Publishers, 1988.

[KOT88] Koton, P.: Reasoning About Evidence in Causal Explanations. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann Publishers, 1988.

[KOT89] Koton, P.: SMARTplan: A Case-Based Resource Allocation and Scheduling System. Proc. Case-Based Reasoning Workshop, 1989, 1989.

[LEA91] Leake, D.: ACCEPTER: a program for dynamic similarity assesment in case-based explanation. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1991

[LEA92] Leake, D.: Constructive Similarity Assessment: using stores cases to define new situations. Proceedings of the Fourteenth Annual conference of the cognitive science society, pp. 313-318, 1992.

[LEA94] Leake, D.: Towards a computer model of memory search strategy learning. Proc. of the 16 annual conference of the Cognitive Science Society, 1994.

[LEA95] Leake, D.; Kinley, A.; Wilson, A.: Learning to improve case adaption by introspective reasoning and cbr. En Case-Based Reasoning Research and Development, First International Conference. ICCBR-95, Lecture Notes in Artificial Intelligence, Springer Verlag, 1995.

[LEH87] Lehnert, W.: Case-based problem solving with a large knowledge base of learned cases. Proc. of the sixth National Conference on Artificial Intelligence. Morgan Kaufmann, Publishers, 1987

[LON94] Long, D.; Garigliano, R.: Reasoning by analogy and causality. Ellis Horwood series in Artificial intelligence, 1994.

[LLU95] Lluís, A- Plaza, E.: Reflection in Noos. An object-centered representation languaje for knowledge modelling. Accesible via url: http://www.iiia.csic.es 1995

[MAR89] Martin, C.: Indexing using complex Features. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[MCG89] McGraw, K.: Knowledge Acquisition. Principles and Guidelines. ed. Prentice-Hall, 1989.

[MIY95] Miyashita, K.: Case-Based Knowledge acquisition for schedule optimization. ARtificial Intelligence in Engineering, 9, 1995.

[NAV88] Navinchandra, D.: Case based reasoning in CYCLOPS, a design problem solver. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1988.

[OKE93] O'Keefe, R-O'Leary, D.: Expert system verification and validation: a survey and a tutorial. Artificial Intelligence Review, 7, 1993.

[OWE88] Owens, C.: Domain-independent prototype cases for planning. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1988.

[OWE89] Owens, C.: Plan transformations as abstract indices. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[OWE93] Owens, C.: Integrating feature extraction and memory search. Machine learning,

10, 1993.

[PAZ89] Pazzani, M.: Indexing strategies for goal specific retrieval of cases. Case-Based Reasoning Workshop. Morgan Kaufmann Publishers, 1989.

[PER89] Perkins, A; Laffey, T.; Pecora, D.; Nguyen, T.: Knowledge Base verification, 1987. En Topics in Expert Systems Design. G. Guida and C. Tasso Editors. North Holland. 1989

[PLA95] Plaza, E.: Cases as Terms: a feature term approach to the structured representation of cases. En Case-Based Reasoning Research and Development, First International Conference. ICCBR-95, Lecture Notes in Artificial Intelligence, Springer Verlag, 1995. Accesible via url: http://www.iiia.csic.es

[PORT89] Porter, B.: Similarity Assessment: computation vs. representation. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[PORI95] Portinale, L; Torasso, P.: ADAPtER: an integrated diagnostic system combining Case-Based and Abductive reasoning. En Case-Based Reasoning Research and Development, First International Conference. ICCBR-95, Lecture Notes in Artificial Intelligence, Springer Verlag, 1995.

[RAM89] Ram, A.: Incremental learning of paradigmatic cases. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[RAM93] Ram, A., 1993: Indexing, Elaboration and Refinement: Incremental Learning of Explanatory Cases. Machine Learning, 10, 201-248.

[RAM94] Ram, A.- Leake, D.: A framework for goal driven learning. Proc. of the 1994 AAAI spring symposium on goal-driven learning. pp 1-11, 1994

[RED90] Redmon, M.: Distributed cases for Case-Based Reasoning; Facilitating Use of Multiple Cases. Cognitive Modeling. pgs 304-309, 1990

[RIC95] Ricci, F.; Avesani, P.: Learning a local similarity metric for Case-Based Reasoning. En Case-Based Reasoning Research and Development, First International Conference. ICCBR-95, Lecture Notes in Artificial Intelligence, Springer Verlag, 1995.

[RIS88] Rissland, E.- Ashley, K.: Credit assignment and the problem of competing factors in case-based reasoning. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1988

[RIS91] Rissland, E.- Skalak, D.: CABARET: rule interpretation in a hybrid architecture. Int. J. Man-Machine Studies, 34, 1991

[ROD94] Rodríguez, A.; Martín, F.: Arquitetura de los Sistemas de Razonamiento Basado en Casos. Technical Report. Universidad de Murcia, 12/94.

[ROD96] Rodríguez, A.; Martín, F.: Knowledge-Based Systems Development. Cybernetics and Systems: An international jounal. 27/2, 1996.

[RUB88] Rubi, D.- Kibler, D.: Exploration of Case-based Problem Solving. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1988

[SAN94] Santamaría, J.- Ram, A.: Systematic Evaluation of Design Decisions in CBR Systems. AAAI Case-Based Reasoning Workshop, WA, 1994.

[SCH82] Schank, R.: Dinamic Memory. A theory of reminding in computers and people. Cambridge University Press, 1982.

[SCH86] Schank, R.: Explanation Patterns, Lawrence Erlbaum Associates, Inc. Hillsdale, NY, 1986.

[SEI94] Seifert, C- Hammond, K.-Converse, T.-McDougal, T.- Vanderstoep S.: Case-Based Learning: Predictive Features in Indexing. Machine Learning 16, 37-56, 1994.

[SHA89] Shanahan, Murray: Search, Inference amd Dependencies in Artificial Intelligence. ed. Ellis Horwood, 1989.

[SIM89] Simoudis, E.- Miller, J.: Validated Retrieval in Case-Based Reasoning. Cognitive Modelling, pp 310- 315, 1989

[SIM91] Simoudis, E.- Miller, J.: The application of CBR to help desk applications. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann Publishers, 1991.

[SYC89] Sycara, K.- Navinchandra, D.: Index transformation and generation for case retrieval. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[SYC91] Sycara, K.- Navinchandra, D.: Influences: a thematic abstraction for creative use of multiple cases. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann Publishers, 1991.

[SYC93] Sycara, K.: Machine learning for intelligent support of conflict resolution. Decision support systems 10. pp 121-136, 1993.

[THA89] Thagard, P,- Holyoak, K.: Why indexing is the wrong way to think about analog retrieval. Case-Based Reasoning Workshop. Morgan Kaufmann Publishers, 1989.

[THO89] Thompson, K- Langley, P.: Organization andretrieval of comosite concepts. Case-Based Reasoning Workshop. Morgan Kaufmann Publishers, 1989.

[VAR93] Vargas, E.- Raj, S.: Developing maintainable expert systems using case-based reasoning. Expert Systems, Nov., vol 10. N°4, 1993.

[WALL88] Wall, R.- Donahue, D.- Hill, S.: The use of domain semantics for retrieval and explanation in Case-Based Reasoning. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1988.

[WALT89] Waltz, D.: Is indexing used for retrieval?. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[WAT94] Watson, I.; Marir, F.: Case-Based reasoning: a review. The Knowledge Engineering Review, Vol. 9.4 pp 327-354, 1994.

[WEB95] Weber, L.; Miranda, R.; Khator, S.: Case-Based Reasoning for cash flow forecasting using fuzzy retrieval. En Case-Based Reasoning Research and Development, First International Conference. ICCBR-95, Lecture Notes in Artificial Intelligence, Springer Verlag, 1995.

[WHI89] Whitaker, L.: Using qualitative or multi attribute similarity to retrieve useful cases form a case base. Proc. Case-Based Reasoning Workshop. Morgan Kaufmann, Publishers, 1989.

[WOL95] Wolverton, M.. An investigation of marker-passing algorithms for analogue retrieval. En Case-Based Reasoning Research and Development, First International Conference. ICCBR-95, Lecture Notes in Artificial Intelligence, Springer Verlag, 1995.

[YAN93] Yan, K.: Applying Conceptual clustering to knowledge-bases construction. Decision Support Systems 10, 173-198, 1993.