UNIVERSIDAD DE MÁLAGA

Escuela Técnica Superior de Ingeniería Informática

Departamento de Lenguajes y Ciencias de la Computación

Programa de Doctorado en Tecnologías Informáticas

TESIS DOCTORAL

De la Información al Conocimiento

Aplicaciones basadas en implicaciones y computación paralela

Autor: D. Fernando Benito Picazo

Directores: Dr. D. Manuel Nicolás Enciso García-Oliveros
Dr. D. Carlos Manuel Rossi Jiménez

Málaga, 2018

UNIVERSIDAD DE MÁLAGA

Escuela Técnica Superior de Ingeniería Informática

Departamento de Lenguajes y Ciencias de la Computación

Dr. D. Manuel Nicolás Enciso García-Oliveros, Catedrático de Escuela Universitaria del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga,
Dr. D. Carlos Manuel Rossi Jiménez, Catedrático de Escuela Universitaria del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga

HACEN CONSTAR QUE:

D. Fernando Benito Picazo, Ingeniero en Informática, ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, bajo nuestra dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulado:

De la Información al Conocimiento

Aplicaciones basadas en implicaciones y computación paralela

Revisado el presente trabajo, estimamos que puede ser presentado al tribunal que ha de juzgarlo, y autorizamos la presentación de esta Tesis Doctoral en la Universidad de Málaga.

Málaga, Noviembre de 2018

Dr. Manuel N. Enciso García-Oliveros Dr. Carlos M. Rossi Jiménez

A mi querida Aurora,

a mi padre y a mi madre,

a mi hermano,

por el apoyo y la confianza que

siempre me habéis demostrado.

¡Y a la alegría de la casa, la Elsita!

En todo objetivo conseguido hay siempre una especial tristeza,
en el conocimiento de que una meta largamente deseada
se ha logrado al fin, y que la vida tiene entonces que ser
moldeada y encaminada en busca de nuevos fines.

La Ciudad y las Estrellas

A. C. Clarke

Agradecimientos

A menudo he escuchado que esta es la parte que más difícil resulta redactar; coincido con ello, pero también tengo claro que es la que más he disfrutado escribiendo.

Ahora que ya se termina esta etapa de mi vida con la consecución de esta tesis doctoral, es momento de hacer balance y de agradecer a las personas que han intervenido en este gran logro; porque es también gracias a ellos.

En primer lugar, quiero agradecer a mis queridos directores, Dr. Manuel Enciso y Dr. Carlos Rossi, la dirección de la tesis y todo lo que me habéis enseñado, pero en especial, vuestro trato, con el que siempre me habéis hecho sentir uno de vosotros, y donde la acogida siempre ha sido la mejor; por todo ello, gracias.

Manolo, me has dirigido: el proyecto fin de carrera de la ingeniería técnica de sistemas, el proyecto fin de carrera de la ingeniería superior, el trabajo fin de máster, y ahora, la tesis doctoral. Quiero decirte que me siento orgulloso de ello, y que si tuviera que volver a recorrer todo ese camino, volvería a pedirte que estuvieras de nuevo a mi lado; por todo ello, gracias.

Carlos, lo primero es decirte que ojalá nos hubiéramos conocido antes. Además de por la dirección de la tesis, también quiero darte las gracias por haber buscado siempre posibilidad de financiación para que pudiera afrontar este periodo de investigación con un respaldo; por todo ello, gracias.

Gracias a mis coautores: Dr. Pablo Cordero, Dr. Ángel Mora, Dr. An-

tonio Guevara, ha sido un placer publicar a vuestro lado; a la Dra. Llanos Mora como tutora de tesis; al Dr. Antonio Vallecillo por buscar un contacto en el momento preciso; al Dr. Darío Guerrero y al Dr. Rafael Larrosa por el apoyo en el Centro de Supercomputación.

Quiero hacer mención especial al Dr. Sergio Gálvez. Sergio, ha sido una gran suerte tenerte cerca para dudas y consejos, y en realidad, por el simple hecho de conversar contigo. Siempre me has apoyado y valorado mucho, y eso significa mucho para mí; por todo ello, gracias.

Ahora, quiero darte las gracias a ti, mi querida Aurora, por ser mi apoyo y ánimo diario e incondicional, por quererme siempre cada día y entender mi insaciable inquietud, y por haber vivido juntos todos estos éxitos que la vida nos está deparando, que seguro, a tu lado, serán muchos más.

Lo primero que quiero decirle a mi padre y a mi madre es: lo he conseguido. Gracias por darme el mejor entorno familiar posible, por vuestro interminable esfuerzo y por inculcarme la vital importancia de la educación y el saber. Ahora es momento de recoger los frutos y disfrutar de lo que hemos conseguido todos juntos.

Gracias también a ti, hermano, por ser la persona que más me ha enseñado en la vida y de quien sigo aprendiendo a cada momento juntos. Gracias por estar siempre disponible y por encauzarme y guiarme por el infinito camino de la Ciencia.

Quiero hacer un guiño a mis profesores de la niñez, D. Justo, D. Julián, D. Luis y D. Andrés, por construir a tan temprana edad, unos sólidos cimientos sobre los que crecer.

Seguramente olvide alguien, no obstante, gracias a toda persona que hasta el día de hoy me haya enseñado cualquier cosa, pues para mí, lo más importante es el *Conocimiento*.

Índice general

Publicaciones

A continuación se expone una lista de los trabajos que han sido publicados como resultado de la investigación llevada a cabo a lo largo de esta tesis doctoral. Estas publicaciones avalan el trabajo realizado poniendo de manifiesto tanto su interés como su validez científica.

Revistas

(I) Fernando Benito-Picazo, Pablo Cordero, Manuel Enciso, Ángel Mora. *Minimal generators, an affordable approach by means of massive computation.* The Journal of Supercomputing, Springer, 2018.

`DOI: 10.1007/s11227-018-2453-z`

Factor de impacto en J.C.R. 2017: 1,532. Posición 43 de 103 (Q2) en la categoría: 'Computer Science, Theory & Methods'.

(II) Fernando Benito-Picazo, Manuel Enciso, Carlos Rossi, Antonio Guevara. *Enhancing the conversational process by using a logical closure operator in phenotypes implications.* Mathematical Methods in the Applied Sciences, John Wiley & Sons Ltd, 2017.

`DOI: 10.1002/mma.4338`

Factor de impacto en J.C.R. 2017: 1,18. Posición 91 de 252 (Q2) en la categoría: 'Mathematics, Applied'.

(III) Fernando Benito-Picazo, Pablo Cordero, Manuel Enciso, Ángel Mora. *Reducing the search space by closure and simplification paradigms. A*

parallel key finding method. The Journal of Supercomputing, Springer, 2016.

DOI: 10.1007/s11227-016-1622-1

Factor de impacto en J.C.R. 2016: 1,349. Posición 52 de 104 (Q2) en la categoría: 'Computer Science, Theory & Methods'.

Congresos Internacionales

- Fernando Benito-Picazo, Pablo Cordero, Manuel Enciso, Ángel Mora. *Closed sets enumeration: a logical approach*. Proceedings of the Seventeenth International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE, 2017. Cádiz, Spain, July 4-8, pp. 287-292, ISBN: 978-84-617-8694-7.

- Fernando Benito-Picazo, Manuel Enciso, Carlos Rossi, Antonio Guevara. *Conversational recommendation to avoid the cold-start problem*. Proceedings of the Sixteenth International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE, 2016. Cádiz, Spain, July 4-8, pp. 184-190, ISBN: 978-84-608-6082-2.

- Fernando Benito-Picazo, Pablo Cordero, Manuel Enciso, Ángel Mora. *Keys for the fusion of heterogeneous information*. Proceedings of the Fifteenth International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE, 2015. Cádiz, Spain, July 6-10, pp. 201-211, ISBN: 978-84-617-2230-3.

- Fernando Benito-Picazo, Pablo Cordero, Manuel Enciso, Ángel Mora. *Increasing the Efficiency of Minimal Key Enumeration Methods by Means of Parallelism*. Proceedings of the 9th International Conference on Software Engineering and Applications, ICSOFT-EA, Vienna, Austria, August 29-31, 2014, pp. 512-517.

DOI: 10.5220/0005108205120517

Workshops

- Fernando Benito-Picazo. *Parallelism in the search of minimal keys from implications using tableaux methods*. Workshop: Lógica, Lenguaje e Información. Dpto. Matemática Aplicada. Unidad de Investigación en Lógica, Lenguaje e Información, Andalucía Tech. Universidad de Málaga, Noviembre 2014.

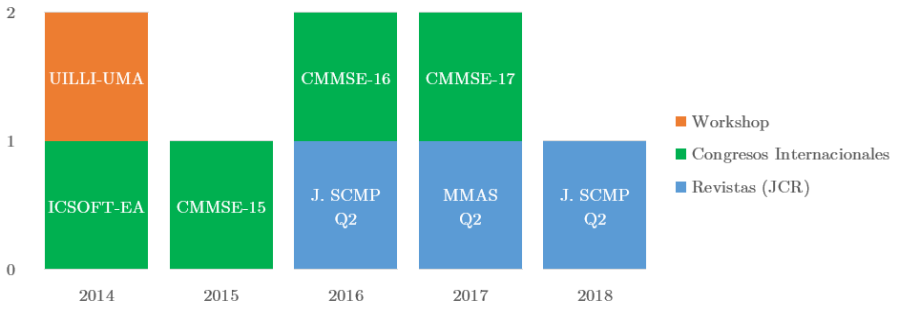

Figura 1: Producción científica

Capítulo 1

Introducción

—Empieza por el principio —dijo el Rey con gravedad— y sigue hasta llegar al final; allí, te paras.

Alicia en el país de las maravillas

L. Carroll

La gestión de la información es uno de los pilares esenciales de la Ingeniería Informática. No es de extrañar, por tanto, que conforme un amplio campo de investigación y conocimiento donde diversas disciplinas como las Matemáticas, la Lógica y la Computación actúen conjuntamente para alcanzar mejores sinergias.

Dentro de este ámbito y con la intención de hacer aportaciones en campos de la Ingeniería Informática como son las bases de datos y los sistemas de recomendación, esta tesis doctoral toma como principal base teórica el Análisis Formal de Conceptos (FCA, por sus siglas en inglés: *Formal Concept Analysis*), y más concretamente, una de sus herramientas fundamentales: los conjuntos de implicaciones. La gestión inteligente de estos elementos mediante técnicas lógicas y computacionales confieren una alternativa para superar obstáculos en los campos mencionados.

FCA es una teoría matemática y una metodología que permite derivar una jerarquía de conceptos a partir de una colección de objetos, sus atribu-

tos y las relaciones entre ellos. De esta forma, el propósito es poder representar y organizar la información de manera más cercana al pensamiento humano sin perder rigor científico. En este sentido se enmarca la cita de Rudolf Wille: "*El objetivo y el significado del FCA como teoría matemática sobre conceptos y sus jerarquías es apoyar la comunicación racional entre seres humanos mediante el desarrollo matemático de estructuras conceptuales apropiadas que se puedan manipular con la lógica.*" [127].

El término FCA fue acuñado por Wille en 1984 culminando años más tarde con la publicación más citada al respecto en colaboración con Bernhard Ganter [46]. Desde entonces, FCA se ha aplicado con éxito en diferentes disciplinas, como por ejemplo: biología celular [39], genética [119], ingeniería del software [65, 95], medicina [104], derecho [82], etc.

FCA parte de una representación de conjuntos de objetos y atributos por medio de tablas de datos. Estas tablas se denominan contextos formales y representan las relaciones binarias entre esos objetos y atributos. A partir de ahí, se generan dos herramientas básicas para representar el conocimiento: los retículos de conceptos y los conjuntos de implicaciones. Dichas herramientas además, son representaciones equivalentes del conocimiento descrito en el contexto formal.

Desde hace años, existen en la literatura estudios [67, 91] donde se han investigado y comparado diferentes algoritmos para obtener el retículo de conceptos a partir de un conjunto de datos (en adelante *dataset* por su nomenclatura habitual en el campo). Muchos de ellos toman como base uno de los algoritmos más conocidos a tal efecto, el denominado por Wille y Ganter como *NextClosure* [46].

Por otro lado está el conjunto de implicaciones. Las implicaciones pueden considerarse *grosso modo* como reglas del tipo *si-entonces*, que representan un concepto muy intuitivo: cuando se verifica una premisa, entonces se cumple una conclusión. Esta idea básica se utiliza con diferentes interpretaciones en numerosos campos de conocimiento. Así, en la teoría relacional se interpretan como dependencias funcionales (DFs) [22], en FCA como implicaciones [46], etcétera.

No obstante, también existen ciertas desventajas a la hora de trabajar

con retículos e implicaciones, de hecho, la propia extracción del conjunto completo de implicaciones de un *dataset* es una tarea que presenta una complejidad exponencial, sin embargo, es conveniente destacar que no es competencia de este trabajo el estudio de técnicas de extracción de implicaciones (lo cual es más una tarea de minería de datos), sino que la intención es partir del conjunto de implicaciones para trabajar con él. A este respecto, se pueden consultar trabajos ampliamente citados en la literatura en relación a la extracción de implicaciones a partir de *datasets* [57, 131].

Trabajar con conjuntos de implicaciones permite utilizar técnicas de razonamiento automático basadas en la lógica. Este hecho fundamenta el objetivo de esta tesis doctoral, que principalmente consiste en, utilizando los conjuntos de implicaciones, aplicar mecanismos lógicos para realizar un tratamiento eficiente de la información.

Como se verá en el Capítulo 2, la aproximación a través de la lógica es posible gracias a sistemas axiomáticos correctos y completos como los axiomas de Armstrong [4] y la Lógica de Simplificación [26] (SL, por sus siglas en inglés: *Simplification Logic*). Estos métodos aplicados sobre conjuntos de implicaciones se utilizan en esta tesis doctoral sobre tres áreas de investigación: claves minimales, generadores minimales y sistemas de recomendación conversacionales.

Se anticipa que, en cada uno de los casos, se va a aprovechar la información subyacente al conjunto de implicaciones para realizar novedosas aproximaciones que permitan abordar problemas presentes en esos ámbitos. Los resultados obtenidos se sustentan por una amplia gama de experimentos, en los cuales se ha utilizado tanto información real como sintética (información generada de forma aleatoria) y donde la computación paralela llevada a cabo en entornos de supercomputación ha desempeñado un papel crucial. Como se verá más adelante, para el primer caso, el trabajo se centra en el uso de DFs mientras que para el segundo y tercero el núcleo son implicaciones.

Dicho esto, se retoma el texto pasando a introducir los tres campos de aplicación donde se han utilizado los conjuntos de implicaciones.

1.1. Claves Minimales

El concepto de clave es fundamental en cualquier modelo de datos, incluyendo el modelo de datos relacional de Codd [24]. Una clave de un esquema relacional está compuesta por un subconjunto de atributos que identifican a cada uno de los elementos de una relación. Representan el *dominio* de una determinada función cuya *imagen* es la totalidad del conjunto de atributos. Así, en un esquema de bases de datos relacional, una clave permite identificar cada fila de una tabla, impidiendo que exista más de una fila con la misma información y puede representarse por medio de una DF [24] hacia todo el conjunto de atributos. Debido a ello en los sistemas gestores de bases de datos, las restricciones de clave son implementadas usando restricciones de unicidad (*unique*) sobre el subconjunto de atributos que forman la clave.

Las DFs especifican una relación entre dos subconjuntos de atributos, e.g. A y B, representada como $A \rightarrow B$, que asegura que para cualesquiera dos tuplas de una tabla de datos, si los valores de sus atributos de A coinciden, entonces también han de coincidir los de B. Si bien la noción de DF se verá con mayor detalle en la Sección 2.2, se adelanta el siguiente ejemplo básico 1.1.1 tanto para mostrar ejemplos de DFs como para ilustrar el concepto de clave.

Ejemplo 1.1.1. *Supongamos que disponemos de la siguiente tabla con información que relaciona títulos de películas, actores, países, directores, nacionalidad, valoración y años de estreno:*

Título	*Año*	*País*	*Director*	*Nacionalidad*	*Actor*	*Valoración*
Pulp Fiction	*1994*	*USA*	*Tarantino*	*USA*	*J. Travolta*	*8*
Pulp Fiction	*1994*	*USA*	*Tarantino*	*USA*	*U. Thurman*	*9*
Pulp Fiction	*1994*	*USA*	*Tarantino*	*USA*	*S. Jackson*	*8*
King Kong	*2005*	*NZ*	*Jackson*	*NZ*	*N. Watts*	*9*
King Kong	*2005*	*NZ*	*Jackson*	*NZ*	*J. Black*	*6*
King Kong	*1976*	*USA*	*Laurentiis*	*IT*	*J. Lange*	*7*
King Kong	*1976*	*USA*	*Laurentiis*	*IT*	*J. Bridges*	*6*
Django Unchained	*2012*	*USA*	*Tarantino*	*USA*	*J. Foxx*	*8*
Django Unchained	*2012*	*USA*	*Tarantino*	*USA*	*S. Jackson*	*9*
Blade Runner	*1982*	*USA*	*Scott*	*UK*	*H. Ford*	*9*
Blade Runner	*2017*	*USA*	*Villeneuve*	*CAN*	*H. Ford*	*6*

De esta información, podemos extraer el siguiente conjunto de DFs:

$\{Titulo, Año \rightarrow Pais, Director;\ Director \rightarrow Nacionalidad\}$.

Esta tabla tiene una única clave: {*Título, Año, Actor*} *que corresponde con el conjunto de atributos necesario para identificar cualquier tupla de la relación.*

La identificación de las claves de una determinada relación es una tarea crucial para muchas áreas de tratamiento de la información: modelos de datos [112], optimización de consultas [63], indexado [78], enlazado de datos [90], etc. Como muestras de esta importancia, es posible encontrar numerosas citas en la literatura, entre las que se pueden destacar las siguientes. En [113], los autores afirman que: "*la identificación de claves es una tarea fundamental en muchas áreas de la gestión moderna de datos, incluyendo modelado de datos, optimización de consultas (proporciona un optimizador de consultas con nuevas rutas de acceso que pueden conducir a mejoras sustanciales en el procesado de consultas), indexación (permite al administrador de la base de datos mejorar la eficiencia del acceso a los datos a través de técnicas como la partición de datos o la creación de índices y vistas), detección de anomalías e integración de datos*". En [94] los autores delimitan el problema manifestando: *"establecer enlaces semánticos entre los elementos de datos puede ser realmente útil, ya que permite a los rastreadores, navegadores y aplicaciones combinar información de diferentes fuentes."*.

Como refleja el contenido de esta sección, es evidente la importancia manifiesta de averiguar las claves de una relación, sin embargo, esta labor no está exenta de dificultades, por ello, el trabajo realizado en esta parte de la tesis ha consistido en proponer, diseñar e implementar métodos para afrontar el problema de la búsqueda de claves, el cual se presenta a continuación.

El Problema de la Búsqueda de Claves

El problema de la búsqueda de claves consiste en encontrar todos los subconjuntos de atributos que componen una clave minimal[1] a partir de un conjunto de DFs. Es un campo de estudio con décadas de antigüedad como puede observarse en [106], o en [40], donde las claves se estudiaron dentro del ámbito de la matriz de implicaciones.

El cálculo de todas las claves minimales representa un problema complejo. En [74, 133] se incluyen resultados interesantes acerca de la complejidad del problema; los autores demuestran que el número de claves está limitado por el factorial del número de dependencias, por tanto, no existe un algoritmo que resuelva el problema en tiempo polinómico. En definitiva, es un problema NP-completo decidir si existe una clave de tamaño a lo sumo k dado un conjunto de DFs.

Por otro lado, en [27], los autores muestran cómo el problema de las claves minimales en las bases de datos tiene su análogo en FCA, donde el papel de las DFs se trata como implicaciones de atributos. En ese artículo, el problema de las claves minimales se presentó desde un punto de vista lógico y para ello, se empleó un sistema axiomático, que los autores denominaron SL_{FD} (por sus siglas en inglés: *Simplification Logic for Functional Dependencies*) [26], para gestionar las DFs y las implicaciones.

Las principales referencias sobre el problema de la búsqueda de claves apuntan al trabajo de Lucchesi y Osborn en [74] donde presentan un algoritmo para calcular todas las claves. Por otro lado, Saiedian y Spencer [105] presentaron un algoritmo usando grafos con atributos para encontrar todas las claves posibles de un esquema de base de datos relacional. No obstante, demostraron que sólo podía aplicarse cuando el grafo de DFs no estuviera fuertemente conectado. Es reseñable también el trabajo de Zhang [136] en el cual se utilizan mapas de Karnaugh [62] para calcular todas las claves. Existen más trabajos sobre el problema del cálculo de las claves minimales

[1] Se acuña el término *minimal* para referirnos a una clave en la que todos y cada uno de los atributos que la forman son imprescindibles para mantener su naturaleza de clave, es decir, no contiene ningún atributo superfluo.

como son [113, 128].

Algoritmos para el Cálculo de Claves

El objetivo de esta parte de la tesis se centra en los algoritmos de búsqueda de claves basados en la lógica, y más específicamente, en aquellos que utilizan el paradigma de tableaux [88, 103] como sistema de inferencia.

De forma muy general, se puede decir que los métodos tipo tableaux representan el espacio de búsqueda como un árbol, donde sus hojas contienen las soluciones (claves). El proceso de construcción del árbol comienza con una raíz inicial y desde allí, mediante la utilización de unas reglas de inferencia, se generan nuevas ramas del árbol etiquetadas con nodos que representan instancias más simples del nodo padre. Debido a esta característica, las comparaciones entre estos métodos se pueden realizar fácilmente ya que su eficiencia va de la mano del tamaño del árbol de búsqueda generado. La mayor ventaja de este proceso es su versatilidad, ya que el desarrollo de nuevos métodos se reduce en gran parte a cambiar las reglas de inferencia.

Esto conduce a un punto de partida fundamental, los estudios de R. Wastl (Universidad de Wurzburg, Alemania) [124, 125] donde se introduce por primera vez un sistema de inferencia de tipo Hilbert para averiguar todas las claves de un esquema relacional. Básicamente, se parte de una raíz para cuyo cálculo se ha aplicado una regla de inferencia $\mathbb{K}1$ y a partir de ahí, se van construyendo las diferentes ramas del árbol mediante la aplicación de una segunda regla de inferencia $\mathbb{K}2$ al conjunto de DFs (véase [124, 125] para más detalles).

Siguiendo esta línea, en [25] los autores abordan el problema de la búsqueda de claves utilizando un sistema de inferencia basado en la lógica SL_{FD} [26], demostrando como el árbol del espacio de búsqueda que se genera lleva a sobrepasar las capacidades computacionales de ordenadores corrientes hoy día, incluso para problemas pequeños. En [26] los autores muestran la equivalencia entre SL_{FD} y los axiomas de Armstrong [4] junto con un algoritmo para calcular el cierre de un conjunto de atributos.

Más tarde, en [27], los autores introdujeron el método SST (por sus siglas en inglés, *Strong Simplification Tableaux*) para calcular todas las claves minimales usando una estrategia de estilo tableaux, abriendo la puerta a incorporar el paralelismo en su implementación.

El método SST está basado en la lógica de simplificación y sus equivalencias, añadiendo además, el test de minimalidad para aumentar la eficiencia. De esta forma, SST evita la apertura de ramas adicionales del árbol, por lo que el espacio de búsqueda se vuelve más reducido, logrando un gran rendimiento en comparación con sus predecesores como puede comprobarse en el amplio estudio realizado sobre el método en [10].

Por otro lado, el nuevo operador de cierre definido en [87] tiene una característica fundamental que lo convierte en una novedosa alternativa frente a los métodos clásicos [75] y es la siguiente. Además del conjunto de atributos que se deriva de la aplicación del operador de cierre al conjunto de implicaciones, el método proporciona un subconjunto Σ' de implicaciones del conjunto Σ original que engloba la información que ha quedado fuera del cierre.

Tomando como base esos trabajos anteriores y con el apoyo del sistema axiómatico de la lógica SL_{FD} (véase Sección 2.3.2), en esta tesis se presenta un nuevo método llamado *Closure Keys (CK)*. Este nuevo método incorpora un mecanismo eficiente de poda que utiliza el método de cierre basado en SL_{FD} (ver Sección 2.4.1) para mejorar el rendimiento del método SST.

Una propiedad muy interesante de los métodos basados en tableaux (como lo son los métodos SST y CK) es la generación de subproblemas independientes los unos de los otros a partir del problema original. De esta forma, se alcanza otro objetivo fundamental de esta tesis, que consiste en utilizar las técnicas lógicas sobre una implementación paralela de los métodos de búsqueda de claves que, mediante el uso de recursos de supercomputación, permitan alcanzar resultados en un tiempo razonable.

Por nuestra parte, en [11] ya se presentó una primera aproximación a la paralelización del método de Wastl [124, 125] y el algoritmo de claves [25], donde se muestra cómo el paralelismo puede integrarse de forma natural en los métodos basados en tableaux. Siguiendo la línea de estos trabajos,

en esta tesis se ha llevado a cabo el estudio y diseño de los métodos SST y CK, y posteriormente, se han desarrollado también las implementaciones de los algoritmos en sus versiones secuenciales y paralelas, basándose estas últimas en el paradigma *MapReduce* [35].

Para la labor de computación de alto rendimiento, se ha trabajado intensamente con el Centro de Supercomputación y Bioinnovación de la Universidad de Málaga[2]. La posibilidad de tratar con este centro ha proporcionado dos beneficios fundamentales: por un lado, se ha alcanzado una elevada pericia para trabajar en entornos de computación de alto rendimiento (HPC, por sus siglas en inglés: *High Performance Computing*) y para realizar implementaciones que aprovechen una alta cantidad de recursos, y por otro lado, ha permitido obtener resultados empíricos sobre experimentos utilizando estrategias paralelas que han desembocado en contribuciones científicas [11, 13] y que habría sido imposible conseguir en la actualidad sin contar con tales recursos computacionales.

Básicamente, el algoritmo paralelo de búsqueda de claves se divide en dos partes principales. Utiliza una primera fase en la que se realiza una expansión del árbol de búsqueda trabajando sobre el problema original y aplicando sucesivamente las reglas de inferencia y el algoritmo del cierre lógico, pero llegando únicamente hasta un cierto nivel de árbol, es decir sin alcanzar todavía las claves en las hojas del árbol. A partir de ese momento, se tiene un árbol de búsqueda parcial en el que cada nodo constituye un problema equivalente al original pero simplificado. A continuación interviene la segunda etapa del algoritmo y la computación paralela en la que cada nodo de ese nivel del árbol, se resuelve en paralelo mediante el uso de un elevado número de procesadores, es decir, aplica el mismo algoritmo de búsqueda de claves, pero ahora ya sí, hasta alcanzar las hojas del árbol, es decir, las soluciones del problema.

Existen numerosos factores a tener en cuenta a la hora de aplicar el algoritmo paralelo, de entre los cuales, el más importante es el valor de corte o parada de la primera etapa (en adelante BOV por sus siglas en

[2]http://www.scbi.uma.es/

inglés, *Break-Off Value*). Determinar este valor es un punto muy sensible del problema, pues de él depende el aprovechamiento general de los recursos en la aplicación del paralelismo [13]. Esto se debe a que existe la necesidad de elegir un BOV de forma que la primera fase del algoritmo no requiera una cantidad excesiva de tiempo de ejecución, pero al mismo tiempo, que se genere la suficiente información para poder maximizar el rendimiento de la segunda fase, la de computación paralela.

Para contrastar la aportación del algoritmo, se han realizado una considerable cantidad de pruebas de rendimiento, las cuales necesitan llevarse a cabo en entornos de supercomputación y cuyos resultados pueden consultarse en [13]. Así, se ha demostrado que el algoritmo diseñado es claramente susceptible de ejecutarse utilizando una implementación paralela. Se puede comprobar como se consiguen resultados en tiempos razonables incluso en los casos en los que la cantidad de información de entrada es considerable y en los que los métodos secuenciales no son capaces de finalizar.

1.2. Generadores Minimales

Como se ha mencionado anteriormente, una forma de representar en FCA el conocimiento es el retículo de conceptos. Esta representación otorga una visión global de la información con un formalismo muy sólido, abriendo la puerta para utilizar la teoría de retículos como una metateoría para gestionar la información [16].

Los conjuntos cerrados son la base para la generación del retículo de conceptos ya que éste puede ser construido a partir de aquellos, considerando la relación de subconjuntos como la relación de orden. En este punto nace el concepto de generadores minimales como representaciones canónicas de cada conjunto cerrado [46].

Los generadores minimales junto con los conjuntos cerrados son esenciales para obtener una representación completa del conocimiento en FCA. Su relevancia puede apreciarse a través de importantes estudios como [96, 98]. Además, los generadores minimales se han usado como punto clave para generar bases, las cuales constituyen una representación compacta del

conocimiento que facilita un mejor rendimiento de los métodos de razonamiento basados en reglas. Missaoui et al. [83, 84] presentan el uso de generadores minimales para calcular bases que impliquen atributos positivos y negativos cuyas premisas son generadores minimales.

En este aspecto, el trabajo de esta parte de la tesis ha consistido en el estudio y diseño de métodos para la enumeración de todos los conjuntos cerrados y sus generadores minimales a partir del conjunto de implicaciones. El proceso se desarrolla a partir de esta información, y no del *dataset* original, lo cual, hasta donde se ha investigado, no se había hecho previamente.

Métodos para el Cálculo de Generadores Minimales

Los métodos propuestos en esta tesis son una evolución del presentado en [28], donde se utilizó la lógica SL_{FD} como medio para encontrar todos los generadores minimales a partir de un conjunto de implicaciones. Este método trabaja sobre el conjunto de implicaciones aplicando unas reglas de inferencia y construyendo árbol de búsqueda de aspecto similar a los árboles del caso de las claves minimales. No obstante, hay una diferencia esencial en el caso de los generadores minimales y es la siguiente.

Al igual que el algoritmo CK para claves minimales, los métodos propuestos utilizan el cierre SL_{FD}, con lo cual, en cada paso también obtienen un conjunto Σ' reducido de implicaciones, pero ahora además, cada nodo del tableaux que se genera es una solución parcial del problema, la cual se combinará con el resto de soluciones al término de la ejecución del algoritmo para obtener el resultado final, mientras que en el caso de claves minimales, las soluciones se encontraban únicamente en las hojas.

Tras el método presentado en [28] (que los autores denominaron MinGen) se presenta ahora un nuevo método, MinGenPr, que aplica una importante mejora con respecto al anterior. Fundamentalmente, consiste en incorporar un mecanismo de poda, basada en un test de inclusión de conjuntos, que involucra a todos los nodos del mismo nivel, para evitar la enumeración de generadores minimales y cierres redundantes. El propósito de esta poda es verificar la información de cada nodo en el espacio de

búsqueda, evitando la apertura de una rama completa.

Finalmente, se propone un último método, GenMinGen, que generaliza la estrategia de poda anterior al considerar el test de inclusión del subconjunto no sólo con la información de los nodos del mismo nivel, sino también con todos los generadores minimales calculados antes de la apertura de cada rama.

En definitiva, se han estudiado, diseñado e implementado cada uno de estos métodos en su versión secuencial. Para evaluar el rendimiento e ilustrar las mejoras obtenidas al pasar de un método a otro, se han realizado un gran número de pruebas utilizando información sintética e información real procedente de repositorios de datos utilizados comúnmente en investigación, como son los de la Universidad de California, Irvine (UCI)[3].

A la luz de los resultados obtenidos en [14], se aprecia claramente como la estrategia de poda del método MinGenPr hace que su rendimiento supere con creces al anterior MinGen. Estas mejoras pueden verse reflejadas en la reducción del número de nodos del árbol de búsqueda y su consiguiente disminución de los tiempos de ejecución del algoritmo. Respecto al último método, GenMinGen, los resultados de los experimentos son aún más notables, alcanzando reducciones superiores al 75 % en ambas métricas (i.e. número de nodos y tiempos de ejecución) en muchos de los casos.

Generadores Minimales y Paralelismo

Si se pretende trabajar sobre conjuntos de implicaciones con una cantidad de información substancial, surge el mismo problema que en la enumeración de las claves minimales, la capacidad computacional de una máquina convencional actual no es suficiente para solucionar estos problemas en un tiempo razonable. Por tanto, se vuelve a utilizar el paralelismo como estrategia para abordar el problema.

Aunque GenMinGen ha demostrado tener un mejor rendimiento que MinGenPr (y ambos a su vez un mejor rendimiento que MinGen) como

[3] https://archive.ics.uci.edu/ml/datasets.html

demuestran lo resultados obtenidos en [14], sólo se va a desarrollar una versión paralela del método MinGenPr, denominada *MinGenPar*. Esto se debe a que, cuando se usa el método MinGenPr, no existe necesidad de comunicación entre los nodos del árbol, y por tanto, se puede utilizar la misma filosofía paralela de implementación *MapReduce* en dos etapas que se utiliza en el caso de las claves minimales. Sin embargo, cuando se usa el método GenMinGen, es necesario comparar los resultados obtenidos en cada nodo con el conjunto actual de generadores minimales generados hasta el momento. Esto rompe esa filosofía de implementación, donde cada nodo del árbol está destinado a ser resuelto de forma independiente y sin existir comunicación entre cada uno de ellos. No obstante, esta circunstancia es el objetivo de estudio de uno de los trabajos futuros que se proponen en la Sección 6.2.

Finalmente, para verificar el rendimiento y la idoneidad de los métodos en relación a la aplicación de estrategias paralelas, se ha realizado una amplia batería de pruebas tanto sobre información sintética como información real, tal y como se ha explicado anteriormente para el tema de las claves minimales. Además, las pruebas han incluido tareas de estimación del número óptimo de cores a utilizar, así como del valor de corte más apropiado en la etapa primera de los métodos paralelos. Los resultados obtenidos respecto a esta parte de la investigación pueden consultarse en [12] y, especialmente en [14], que constituye uno de los trabajos que avalan esta tesis doctoral.

1.3. Sistemas de Recomendación Conversacionales

La tercera aportación de esta tesis doctoral haciendo uso de los conjuntos de implicaciones se enmarca en el campo de los sistemas de recomendación (SRs).

De forma muy simplificada, se podría considerar que un SR es un sistema inteligente que proporciona a los usuarios una serie de sugerencias personalizadas (recomendaciones) seleccionadas de un conjunto de elementos (ítems). Comúnmente, los SRs estudian las características de cada usuario e

ítem del sistema, y a partir de ahí, mediante un procesamiento de los datos, encuentra un subconjunto de ítems que pueden resultar de interés para el usuario. Una recopilación de las referencias más notables en el campo de los SRs la encontramos en [102].

Desde los primeros trabajos sobre SRs [56, 100], éstos han estado en continua evolución durante los últimos años [1]. Sin embargo, es con la expansión de las nuevas tecnologías cuando han tenido un acercamiento más directo a la mayor parte de la sociedad debido a su capacidad para realizar todo tipo de recomendaciones sobre productos muy populares (libros [31], documentos [97], música [68], turismo [47], películas [43, 54], etc.).

Los SRs constituyen tanto un importante campo de investigación [37, 114], como un elemento indispensable para sólidos entornos comerciales a nivel mundial (Amazon [72], LinkedIn [99], Facebook [117]), lo cual pone de manifiesto la importancia de estos sistemas en la sociedad actual.

Abordar la generación de recomendaciones haciendo uso de FCA es una aproximación existente en la literatura desde hace años. En [36], los autores utilizan FCA para agrupar elementos y usuarios en conceptos para posteriormente, realizar recomendaciones colaborativas según la afinidad con los elementos vecinos. Más tarde, en [109], se introducen un modelo para el filtrado colaborativo basado en FCA para generar correlaciones entre datos a través de un diseño del retículo. Zhang et al. [135] propusieron un sistema basado en similitud agrupando la información contextual en grafos mediante el cual llevar a cabo recomendaciones sobre las interacciones sociales entre usuarios. En [70, 71], se utilizan relaciones difusas e implicaciones ponderadas para especificar el contexto y SL_{FD} para desarrollar un proceso lineal de filtrado que permite a los SRs podar el conjunto original de elementos y así mejorar su eficiencia. Recientemente, en [139] se propone y utiliza un novedoso SR personalizado basado en el retículo de conceptos para descubrir información valiosa de acuerdo con los requisitos e intereses de los usuarios de forma rápida y eficiente. Todos estos trabajos subrayan claramente cómo FCA puede aplicarse con éxito en el campo de los SRs.

Existen numerosos tipos de SRs atendiendo a cómo se generan las re-

comendaciones. Los más extendidos son los de filtrado colaborativo [81] que basan su funcionamiento en las valoraciones de los usuarios a los elementos disponibles; y los sistemas basados en contenido [134] que proporcionan resultados que tengan características similares a otros valorados anteriormente por el usuario. Por otro lado, los SRs basados en conocimiento [77] utilizan un método de razonamiento para inferir la relación entre una necesidad y una posible recomendación.

Los SRs más importantes desde el punto de vista de esta tesis son los denominados conversacionales [50, 69]. Estos SRs se diferencian de los anteriores en el flujo de trabajo que se sigue para generar la recomendación. Este tipo de SR es la estrategia principal para el SR realizado y que ha dado lugar a una de las contribuciones que avalan esta tesis [15]. Se puede consultar una clasificación más detallada en [2, 17].

No obstante, la mejor alternativa consiste en combinar características de diferentes tipos de SRs para generar híbridos que se beneficien de las ventajas de cada uno de ellos [33]. Tal es el caso de este trabajo, en el que se ha realizado un SR híbrido que combina características de los SRs basados en conocimiento, contenido y, principalmente, conversacionales.

La evaluación de las predicciones y recomendaciones es un aspecto fundamental en los SRs [55, 101]. Los SRs requieren medidas de calidad y métricas de evaluación [51] para conocer la calidad de las técnicas, métodos y algoritmos para las predicciones y recomendaciones.

No obstante, dependiendo del SR con el que se trabaje, la evaluación se debe llevar a cabo utilizando aquellas métricas, que por su naturaleza y significado, sean coherentes con el SR que se desea evaluar. En el caso de estudio de esta tesis, dado el SR conversacional desarrollado, una medida adecuada de rendimiento consiste en calcular el número de pasos que se producen en la conversación [80]. Por contra, otras métricas tan populares como son *Precision* y *Recall* [52] no son adecuadas de aplicar en el trabajo de esta tesis porque se obtendría siempre valores máximos en ambas métricas, y la razón es la siguiente. En primer lugar, cualquier ítem de la lista de resultados, verifica los atributos seleccionados ya que la consulta para obtener los ítems resultado contiene esas restricciones. Y en segundo lugar,

a cada paso del diálogo, el sistema devuelve todos los ítems que verifiquen la selección de atributos establecida por el usuario.

Por la misma razón, no existe necesidad de considerar métricas referentes a la exactitud de los resultados ya que el sistema desarrollado no es un modelo de predicción, su funcionamiento está basado en implicaciones y eso asegura la completa de exactitud en las respuestas.

Problemas Comunes y la Maldición de la Dimensión

Si bien es cierto que los SRs están alcanzando una enorme importancia, existen numerosas dificultades que han de afrontarse a la hora de diseñarlos e implementarlos. En la lista de problemas relacionados con los SRs [110] se pueden destacar: el arranque en frío [42, 115], privacidad [44], oveja-negra [49], escasez [53], ataques maliciosos [130, 137], sobreespecialización [73], escalabilidad [59], postergación [116], dimensionalidad [107], etc.

En concreto, en esta tesis se ha orientado el trabajo a abordar este último problema de la dimensionalidad en los SRs. Este problema, también conocido como *the curse of dimensionality phenomenon* [89, 107] aparece cuando es necesario trabajar sobre *datasets* con un alto número de características (variables o atributos). De forma intuitiva, se puede describir de la siguiente manera: cuando hay pocas columnas de datos, los algoritmos de tratamiento inteligente de la información (aprendizaje automático, *clustering*, clasificación, etc.) suelen tener un buen comportamiento. Sin embargo, a medida que aumentan las columnas o características de nuestros ítems, se vuelve más difícil hacer labores predictivas con un buen nivel de precisión. El número de filas de datos necesarias para realizar cualquier modelado útil aumenta exponencialmente a medida que agregamos más columnas a una tabla [79].

Para abordar este problema, se pueden encontrar numerosos trabajos en la literatura [66, 107], especialmente mediante selección de características, que pueden ayudar a descartar aquellas características que no son relevantes

de cara al objetivo buscado. De hecho, estas técnicas ya se aplican en otras áreas como son: algoritmos genéticos o redes neuronales, normalmente centrándose en la aplicación de un proceso automatizado por lotes [123].

Un trabajo interesante en esta área es [60], que establece la idoneidad de los enfoques basados en el conocimiento para los procesos conversacionales. En particular, estos autores utilizan el razonamiento basado en restricciones, en lugar de nuestro enfoque basado en la lógica. Además, este trabajo trata sobre concepto de optimización de consultas, análogo al aplicado en la propuesta de esta tesis. Otro trabajo notable es [118], que comparte el objetivo de disminuir el número de pasos de la conversación. Los autores proponen métricas acerca del número de pasos y tasas de poda, ambas muy similares a las utilizados en este trabajo de tesis. Por otro lado, en [20], los autores demuestran cómo la posibilidad de que sea el usuario el encargado de la selección de atributos genera una ventaja con respecto al hecho de que sea el sistema mismo el encargado de dicha selección. Este hecho respalda el enfoque buscado en esta tesis, en el cual el experto humano guía la conversación y el proceso de selección de características.

Propuesta Desarrollada

El objetivo ha sido abordar el problema de la alta dimensionalidad en los SRs haciendo uso de los conjuntos de implicaciones, a través de un proceso de selección de atributos por parte del usuario mediante el SR híbrido mencionado anteriormente. De esta forma, se ha conseguido reducir el número de pasos necesarios en el diálogo y gestionar favorablemente el problema de la dimensionalidad [15].

Así, el sistema desarrollado se va a centrar principalmente en la primera fase de la recomendación, el filtrado. Para ello, partiendo del conjunto de implicaciones, utiliza la lógica SL_{FD} [26] y, especialmente, el algoritmo del cierre SL_{FD} [87] como motor para facilitar y acelerar la recomendación. Gracias a la aplicación del cierre, el sistema reduce la sobrecarga de información a cada paso del diálogo filtrando aquellos atributos que resulten de la aplicación del cierre a las solicitudes del usuario, consiguiendo una

reducción del número de pasos necesarios en el diálogo.

Se han realizado numerosas pruebas de aplicación para contrastar su validez. Entre ellas, destacan las pruebas utilizando información real sobre enfermedades y fenotipos, como se puede apreciar en una de las contribuciones que avalan este trabajo de investigación [15]. Además, la propuesta desarrollada, al igual que la gran mayoría de los SRs, casa con los conceptos de adaptabilidad y longevidad de los SRs ya que el funcionamiento es independiente de la información de base con la que trabaje, sólo es necesario conocer el conjunto de atributos e implicaciones subyacente a los datos.

1.4. Verificación de los Resultados

Antes de entrar plenamente en el cuerpo de la tesis es necesario hacer una importante aclaración previa con la intención de indicar la manera de certificar la validez de los resultados obtenidos a lo largo de la tesis.

Como se verá en los siguientes capítulos, la labor de investigación se ha centrado en actuar sobre conjuntos de implicaciones, y en ese sentido, para los experimentos realizados se ha contado con unos ficheros de entrada que contenían la información necesaria, y sobre ellos se han obtenido unos resultados. Ahora bien, la forma de verificar que esos resultados son correctos es la siguiente.

En primer lugar y con respecto a los resultados que se presentan en cuanto a claves y generadores minimales, se han realizado numerosos ejercicios en papel intentando buscar casos límites donde la implementación pudiera no ser precisa y se ha comprobado que los resultados obtenidos en papel coincidían exactamente con los calculados por la máquina. Además, dado que para muchos de los experimentos, en los que se llegaban a calcular millones de nodos de un árbol, no era posible comprobar si cada uno de esos cálculos era correcto, para el caso concreto de los experimentos relacionados con claves minimales, la validez de los experimentos viene dada al haber cotejado los resultados con aquellos obtenidos sobre un amplio abanico de experimentos en trabajos anteriores [9, 10] donde su validez quedó demostrada. Además, la validez de los resultados se corrobora igualmente al

alcanzar las mismas soluciones para diferentes métodos cuando cada uno de ellos hace un tratamiento de la información diferente con respecto al otro. En relación a los experimentos con SRs conversacionales, dado que los experimentos no alcanzan números tan costosos de verificar, la validez de los resultados puede demostrarse de forma más asequible siguiendo un desarrollo explícito en papel.

Adicionalmente, y con mayor énfasis en relación a los experimentos que han conllevado la utilización de recursos de supercomputación, cada uno de los experimentos se ha reproducido entre 30 y 50 veces, de forma que los resultados mostrados son fruto de un estudio estadístico posterior más amplio que permite identificar los resultados más fiables, tal y como se sugiere en [48, 129, 138].

Para cada una de las implementaciones realizadas, existía la necesidad de establecer criterios que permitieran evaluar el rendimiento de las pruebas de forma que se pudieran comparar unos métodos con otros. En el caso de los experimentos con claves y generadores minimales, cuando se plantea la idea de la comparación de resultados, lo primero que se pensó fue la medición de los tiempos que necesitaba cada uno de los métodos para obtener los resultados. No obstante, se advirtió que este parámetro está íntimamente ligado a la arquitectura que estemos utilizando para ejecutar el experimento, lo cual hace que el resultado dependa en gran medida de los recursos que se están utilizando y no tanto de la calidad o eficiencia del propio algoritmo. En consecuencia, se oscurecía la utilidad teórica de los resultados obtenidos. Por tanto, se decidió contabilizar la magnitud del árbol y la cantidad de resultados redundantes que se obtienen (véase [9–11]. De esta forma, en el momento de que exista otro método con un código en cualquier otro lenguaje o utilizase recursos *hardware* diferentes que desembocaran en una mejora del tiempo, siempre se puede atender al tamaño del árbol y al número de cálculos redundantes, pudiendo defender si realmente es una mejora en el método o bien, en la ejecución debido a la arquitectura.

A modo de resumen gráfico, la Figura 1.1 muestra un esquema del camino de investigación que se ha seguido en el desarrollo de esta tesis doctoral, apoyado por las principales nociones y referencias.

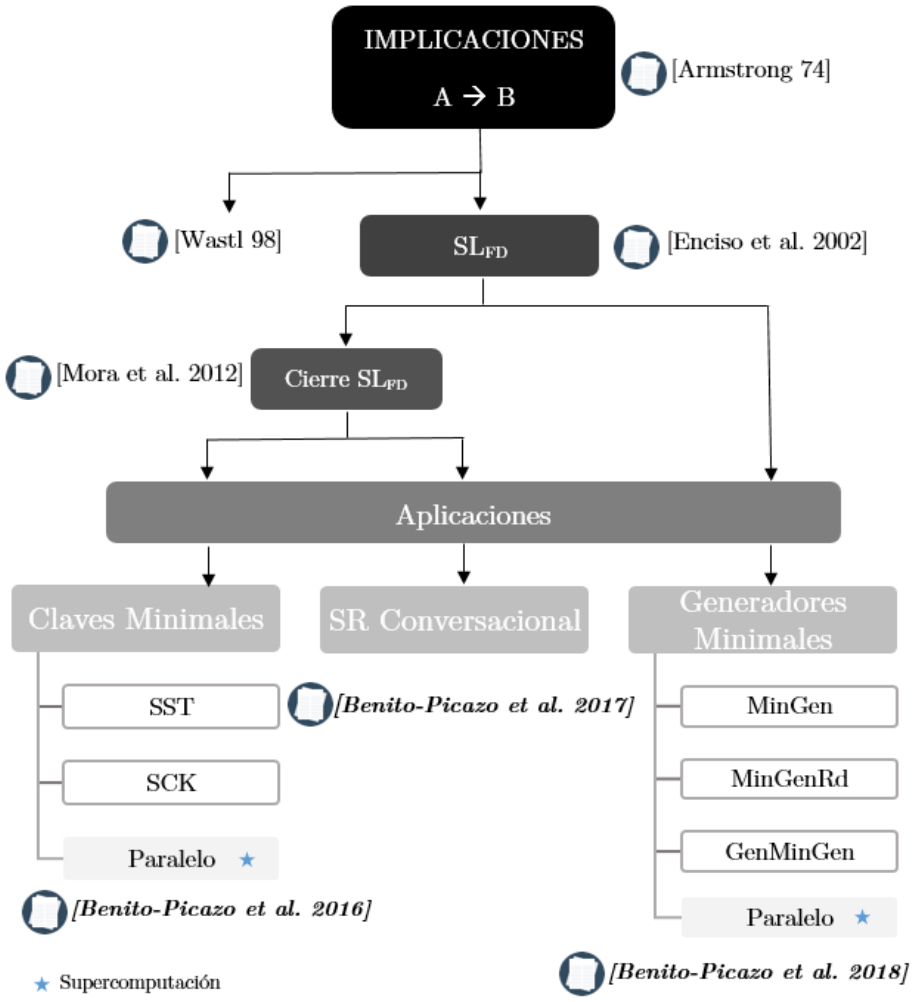

Figura 1.1: Esquema del estado del arte y las contribuciones generadas.

Para finalizar este capítulo, se incluye un último apartado donde se describe brevemente la estructura que presenta el documento incluyendo las publicaciones que avalan esta tesis.

1.5. Estructura de la Tesis

En este primer capítulo de introducción, se han fijado los puntos fundamentales de la tesis, como son: el marco de trabajo sobre el que se va a actuar, las técnicas que se utilizarán y los principales objetivos que se pretenden alcanzar. Concretamente, se ha estipulado la utilización de los conjuntos de implicaciones y las DFs como base del estudio sobre la que aplicar técnicas basadas en la lógica para mejorar el tratamiento de la información.

Tras la introducción, aparece el Capítulo 2, en el que se presentan: el conjunto de nociones principales de FCA para el trabajo realizado sobre generadores minimales y SRs, los aspectos fundamentales de bases de datos y DFs para la investigación sobre claves minimales, las lógicas de implicaciones y los métodos de razonamiento automático utilizados.

A continuación, se presenta el Capítulo 3 en el que se presenta la primera contribución que avala este trabajo de investigación y que corresponde con el trabajo realizado en el campo de la búsqueda de las claves minimales. De forma general, este artículo presenta nuevos métodos para resolver el problema de la inferencia de claves minimales en esquema de datos basándose en la lógica SL_{FD} y el uso de implicaciones. Además, se muestra el funcionamiento de esos métodos y las ventajas obtenidas al aplicar técnicas de computación paralela para poder aplicar los métodos sobre conjuntos de información de un tamaño tal que las técnicas secuenciales no son capaces de gestionar en cuanto a tiempo y recursos necesarios.

Seguidamente, se presenta el Capítulo 4, análogo al anterior pero esta vez para el tema referente a los generadores minimales 3. Este capítulo presenta un segundo artículo en el cual se lleva a cabo un estudio de los métodos de producción de generadores minimales basados en la lógica y el tratamiento de implicaciones. Se comprueba las mejoras de rendimiento de los métodos al aplicar reducciones en el espacio de búsqueda basadas

en estrategias de poda. Al igual que en el caso de las claves minimales, se presenta el funcionamiento de los métodos paralelos para poder tratar con conjuntos de información de tamaño considerable y se incluyen las pruebas realizadas en entornos de supercomputación.

Como último capítulo dedicado a las aplicaciones desarrolladas mediante la gestión de implicaciones se incluye el Capítulo 5. En este capítulo se presenta un novedoso trabajo en el que se desarrolla una aproximación al tratamiento del problema de la dimensionalidad en los SR. Mediante el uso de las implicaciones, la lógica SL_{FD} y el algoritmo del cierre SL_{FD}, se desarrolla un modelo de SR conversacional. Este sistema, es capaz de gestionar el problema de la dimensionalidad reduciendo la sobrecarga de información con la que el usuario debe enfrentarse a la hora de obtener una recomendación. Esta reducción se consigue mediante un filtrado de atributos guiado por la aplicación del cierre. Se demuestra su buen comportamiento mediante su evaluación sobre información real.

Finalmente, la tesis se cierra con el Capítulo 6 dedicado a recopilar las principales conclusiones obtenidas y a proponer caminos por los que seguir ahondando en la investigación. Además, se incluye una relación de las referencias consultadas y los respectivos índices de términos, figuras y tablas.

En aras de la completitud, se incluyen como anexos finales aquellos artículos que han sido publicados a lo largo de este periodo de investigación, que si bien no se utilizan como respaldo para esta tesis doctoral, han sido la semilla y experiencia inicial a partir de la cual se han desarrollado los trabajos que actúan como aval. En la Figura 1.2 se muestra de forma gráfica el contenido de la tesis y se contextualizan las contribuciones publicadas.

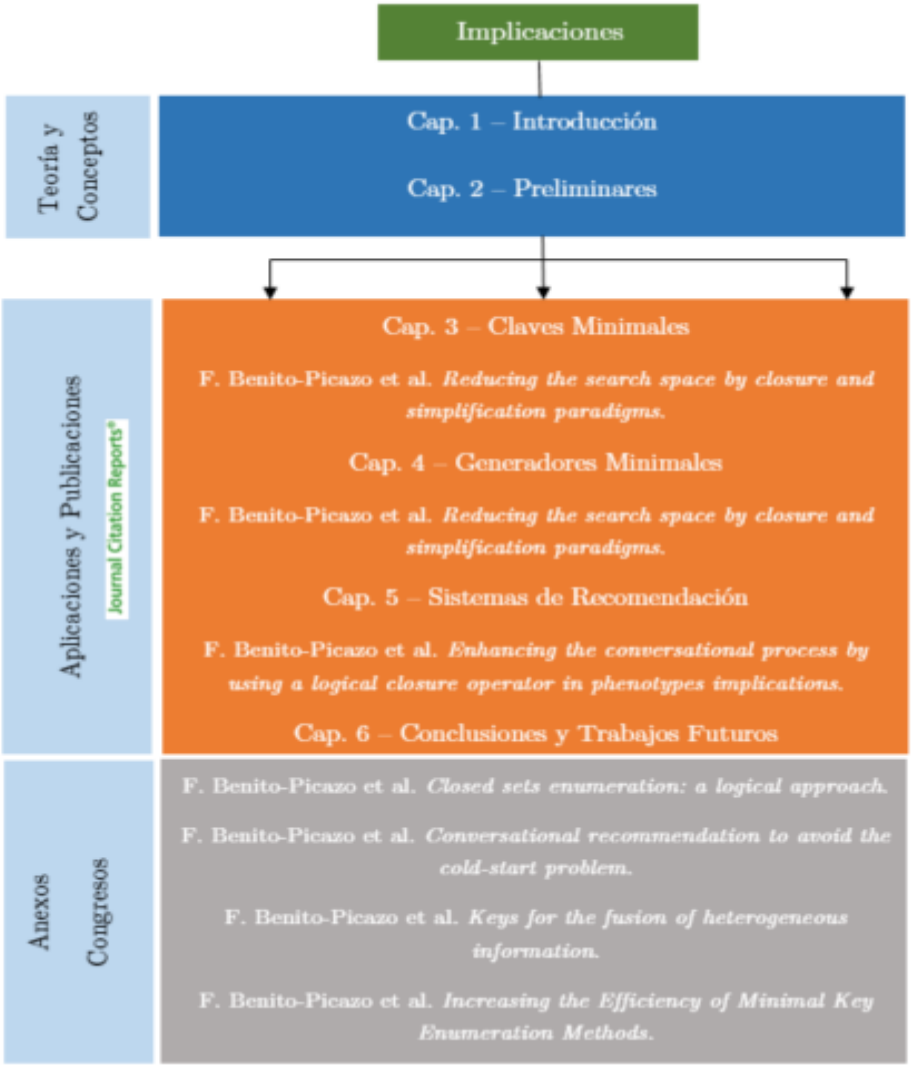

Figura 1.2: Esquema de la estructura de la tesis y las publicaciones.

Capítulo 2

Preliminares

Todo encajaba, con la abrumadora evidencia de una demostración matemática.

Ethan de Athos
L. M. Bujold

A lo largo de este capítulo se van a introducir los principales conceptos, definiciones y resultados sobre los que se sustentan el trabajo de investigación desarrollado. Se parte una vez más de que el objetivo principal de la tesis doctoral es la utilización de los conjuntos de implicaciones para generar soluciones a problemas en el ámbito de FCA, bases de datos y sistemas de recomendación. Se ha diseñado este capítulo con la intención de que el texto sea autocontenido en la medida de lo posible y para ello se ha dividido en cuatro secciones principales tal como muestra la Figura 2.1.

La primera sección presenta las nociones de FCA que serán necesarias para las soluciones propuestas para generadores minimales y sistemas de recomendación. En la segunda, se recopilan los aspectos fundamentales a tener en cuenta para la enumeración de las claves minimales en esquemas relacionales por medio de DFs. La tercera parte se dedica a presentar formalmente las lógicas de implicaciones. La cuarta y última, se centra en el razonamiento automático usando dichas lógicas.

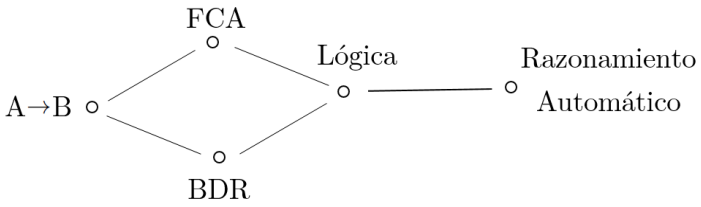

Figura 2.1: Esquema de contenido del Capítulo 2, Preliminares.

2.1. Análisis Formal de Conceptos

De forma general, se puede considerar FCA como un marco para el análisis de información que facilita la extracción de conocimiento y la posibilidad de poder razonar sobre él. La referencia principal de este campo de conocimiento viene de la mano de Wille y Ganter en [46]. Según el propio Wille en [127]: *"El objetivo y significado del análisis formal de conceptos como teoría matemática consiste en apoyar la comunicación racional de las personas, mediante el desarrollo de estructuras de conceptos matemáticamente apropiadas que puedan activarse lógicamente"*.

Como principales fortalezas de FCA cabe mencionar: su sólida base matemática y filosófica, una representación gráfica e intuitiva del conocimiento, avalada en más de 2.000 publicaciones científicas, y aplicaciones en cientos de proyectos [127] en diversos campos de conocimiento, como se ha mencionado en el Capítulo 1.

2.1.1. Contextos Formales

El punto de partida de FCA es un *contexto formal*. Es una noción fundamental para el contenido de esta sección y se define a continuación.

Definición 2.1.1 (Contexto formal)**.** *Un contexto formal es una tripleta* $K = (G, M, I)$ *que consiste en dos conjuntos no vacíos,* G *y* M*, y una relación binaria* I *entre ellos. Los elementos de* G *se llaman objetos del contexto, y los elementos de* M *se llaman atributos del contexto. Para* $g \in G$

y $m \in M$*, escribimos* $< g, m > \in I$ *o* gIm *si el objeto* g *posee el atributo* m*.*

La forma más sencilla de representar un contexto formal es mediante una tabla donde los objetos se sitúan en sus filas y los atributos en las columnas, de forma que un valor en cada celda indica si un objeto $g \in G$ posee un atributo $m \in M$. Hay que tener en cuenta que la definición de contexto formal es deliberadamente muy general. No hay restricciones sobre la naturaleza de los objetos y atributos. Se pueden considerar objetos físicos, personas, números, procesos, estructuras, etc. En realidad, cualquier cosa que sea un *conjunto* en el sentido matemático se puede tomar como el conjunto de objetos o de atributos de algún contexto formal. También es posible intercambiar el papel de los objetos y atributos, es decir, si se tiene que $K = (G, M, I)$ es un contexto formal, igualmente lo será su dual, $K = (M, G, I')$ con $mI'g \Longleftrightarrow gIm$. Además, ni siquiera es necesario que G y M sean disjuntos, de hecho, pueden incluso no ser diferentes.

Para ilustrar un ejemplo de contexto formal y como apoyo al contenido que se presenta en esta sección, se va a utilizar el siguiente ejemplo tomado de [45]. En él, los autores muestran un información sobre los destinos aéreos que oferta un grupo de aerolíneas comerciales.

Ejemplo 2.1.2. *Sea* K *un contexto formal donde el conjunto de objetos* G *comprende todas las líneas aéreas del grupo Star Alliance y el conjunto de atributos* M *muestra sus destinos.*

La relación binaria I *viene reflejada en la Tabla 2.1 y muestra los destinos a los que viaja cada miembro de Star Alliance. Por tanto, se tiene el contexto formal* $K = (G, M, I)$ *en el cual:*

$G = \{$*Air Canada, Air New Zealand, All Nippon Airways, Ansett Australia, The Austrian Airlines Group, British Midland, Lufthansa, Mexicana, Scandinavian Airlines, Singapore Airlines, Thai Airways International, Unites Airlines, VARIG*$\}$

$M = \{$*Latinoamérica, Europa, Canadá, Asia, Oriente Medio, África, México, Caribe, Estados Unidos*$\}$

Tabla 2.1: Ejemplo de contexto formal sobre los destinos aéreos del grupo Star Alliance [45]

	Latinoamérica	Europa	Canadá	Asia	Oriente Medio	África	México	Caribe	Estados Unidos
Air Canada	✓	✓	✓	✓	✓		✓	✓	✓
Air New Zealand		✓		✓					✓
All Nippon Airways		✓		✓					✓
Ansett Australia				✓					
The Austrian Airlines Group		✓	✓	✓	✓	✓			✓
British Midland		✓							
Lufthansa	✓	✓	✓	✓	✓	✓	✓		✓
Mexicana	✓		✓				✓	✓	✓
Scandinavian Airlines	✓	✓		✓		✓			✓
Singapore Airlines		✓	✓	✓	✓	✓			✓
Thai Airways International	✓	✓		✓				✓	✓
Unites Airlines	✓	✓	✓	✓			✓	✓	✓
VARIG	✓	✓		✓		✓	✓		✓

2.1.2. Operadores de Derivación

Dada una selección $A \subseteq G$ de objetos de un contexto formal $K = (G, M, I)$, se desea caracterizar qué atributos de M son comunes a todos estos objetos. Esto define un operador que produce para cada conjunto de objetos $A \subseteq G$, el conjunto A de sus atributos comunes, y dualmente, a partir de un conjunto de atributos, caracterizar el conjunto de atributos que tienen en común a dichos atributos. Estos operadores se denominan operadores de derivación para K.

Definición 2.1.3 (Operadores de derivación)**.** *Dado un contexto formal $K = (G, M, I)$, se definen los operadores de derivación:*

$$(\ )' \colon 2^G \to 2^M \qquad (\ )' \colon 2^M \to 2^G$$

$$A' = \{m \in M \mid g \ I \ m \quad \forall g \in A\} \qquad B' = \{g \in G \mid g \ I \ m \quad \forall m \in B\}$$

Ambas funciones se denotan con el mismo símbolo porque no hay lugar

a confusión.

Ejemplo 2.1.4. *A partir del contexto formal representado en la Tabla 2.1 se pueden aplicar los operadores de derivación y obtener, por ejemplo:*

- $\{Mexicana\}' = \{Latinoamerica,\ Canada,\ Mexico,\ Caribe,\ Estados\ Unidos\}$
- $\{Latinoamerica,\ Europa,\ Asia,\ Africa,\ Estados\ Unidos\}' = \{Scandinavian\ Airlines,\ Lufthansa,\ VARIG\}$

Si A es un conjunto de objetos, entonces A' es un conjunto de atributos, al cual podemos aplicar el segundo operador de derivación para obtener el conjunto de objetos A''. De forma dual, comenzando con un conjunto B de atributos, podemos formar el conjunto de atributos B''.

De la definición anterior se obtiene que:

El operador de cierre $''$ (i.e. aplicar el operador de derivación $'$ y su dual) verifica algunas propiedades interesantes que serán fundamentales para poder desarrollar la teoría formal.

Definición 2.1.5 (Operador de cierre)**.** *Sea $K = (G, M, I)$ un contexto formal, entonces el operador $()'' : 2^M \to 2^M$ es un operador de cierre, es decir, satisface las siguientes propiedades:*

- *Idempotente:* $X'''' = X'' \quad \forall X \in 2^M$
- *Monótona:* $X \subseteq Y \to X'' \subseteq Y'' \quad \forall X, Y \in 2^M$
- *Extensiva:* $X \subseteq X'' \quad \forall X \in 2^M$

Por dualidad, el resultado es igualmente válido para $()'' : 2^G \to 2^G$.

Un conjunto $A \subseteq M$ se denomina *conjunto cerrado* para el operador $''$ si es un punto fijo para $''$, es decir, $A'' = A$. Los conjuntos cerrados permiten definir lo que se denominan *conceptos formales*, los cuales son una noción esencial en FCA y a los que se dedica el siguiente apartado.

Ejemplo 2.1.6. *A partir del contexto formal representado en la Tabla 2.1 se puede obtener el conjunto cerrado de atributos:*

Z = {Latinoamerica, Europa, Canada, Asia, Oriente Medio, Africa, Mexico, Estados Unidos}.

Ya que $Z'' = Z$.

2.1.3. Conceptos Formales

De forma general, un concepto formal permite describir formalmente un hecho del contexto y caracterizar un conjunto de objetos por medio de los atributos que comparten y viceversa.

Así, un par (X, Y) con $X \subseteq G$ y $Y \subseteq M$ es un concepto formal cuando se verifica que:

- Cada objeto en X tiene todos los atributos de Y, y dualmente, cada atributo de Y está presente en cada objeto de X.
- Para cada objeto en G que no está en X, existe un atributo en Y que el objeto no tiene. Dualmente, para cada atributo en M que no está en Y, hay un objeto en X que no tiene ese atributo.

Se introduce ahora la noción formalmente:

Definición 2.1.7 (Concepto formal)**.** *Sea $K = (G, M, I)$ un contexto formal y $A \subseteq G$, $B \subseteq M$. El par (A, B) se denomina concepto formal si $A' = B$ y $B' = A$. El conjunto de objetos A se denomina extensión del concepto (A, B) mientras que el conjunto de atributos B será la intensión del concepto.*

La descripción de un concepto a través de su extensión e intensión es redundante, ya que cada una de las dos partes determina la otra debido a que $B = A'$ y $A = B'$, sin embargo, pueden existir ocasiones en las cuales esta descripción redundante puede ser conveniente [45].

Una alternativa gráfica de identificar los conceptos formales es la siguiente. A partir de la representación del contexto formal, un concepto formal se puede reconocer por medio de un conjunto de elementos que

comparten exactamente las mismas relaciones. De esta forma, en la Tabla 2.2, se ve como, a partir del contexto formal, es posible identificar el concepto formal: {*Air New Zealand, All Nippon Airways*},{*Europa, Asia, Estados Unidos*}.

Tabla 2.2: Extracto del ejemplo de contexto formal sobre los destinos aéreos del grupo Star Alliance 2.1

	Latinoamérica	Europa	Canadá	Asia	Oriente Medio	África	México	Caribe	Estados Unidos
[...]									
Air New Zealand		✓		✓					✓
All Nippon Airways		✓		✓					✓
[...]									

2.1.4. Retículo de Conceptos

Un contexto formal puede tener muchos conceptos formales. El conjunto de todos los conceptos formales de un contexto formal K tiene estructura de *retículo* con la relación de orden que se muestra a continuación.

Si (X_1, Y_1) y (X_2, Y_2) son conceptos, se define un orden parcial, $\leq$, de forma que $(X_1, Y_1) \leq (X_2, Y_2)$ si y sólo si $X_1 \subseteq X_2$, o equivalentemente, si $Y_2 \subseteq Y_1$.

La Figura 2.2 muestra el retículo de conceptos asociado al contexto formal mostrado anteriormente en el Ejemplo 2.1.2. En un diagrama como el de la Figura 2.2, cada nodo representa un concepto formal. Un concepto c_1 es un subconcepto de un concepto c_2 si y solo si hay un camino de descendente desde el nodo que representa c_2 al nodo que representa c_1. El nombre de un objeto g se asocia al nodo que representa el concepto más pequeño que contiene g en su extensión; dualmente, el nombre de un atributo m va asociado al nodo que representa el concepto más grande con m en su intensión.

Se pueden comprobar fácilmente las relaciones que existen en el contexto ya que un objeto g tiene un atributo m si y sólo si el concepto asociado a g es un *subconcepto* del asociado a m. La extensión de un concepto consiste en todos aquellos objetos cuyas etiquetas están asociadas a subconceptos, y, dualmente, la intensión consiste en todos los atributos asociados a *superconceptos*.

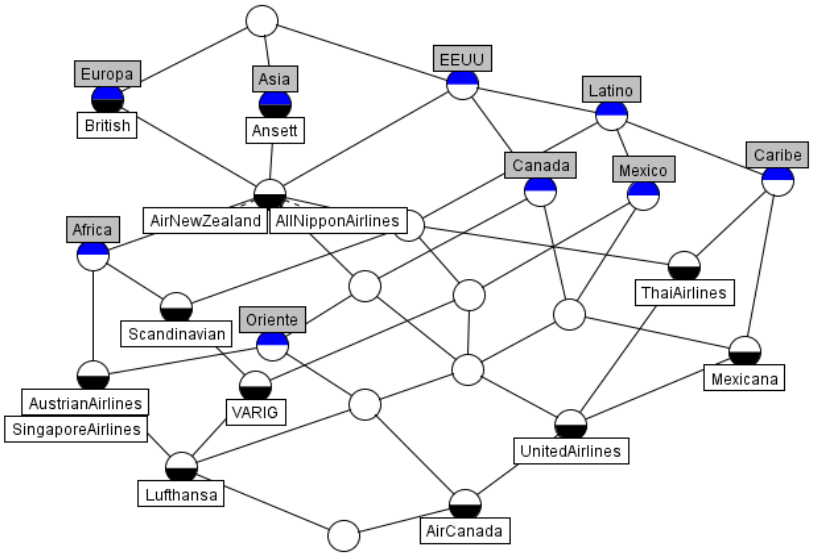

Figura 2.2: Retículo de conceptos asociado al contexto formal 2.1.2

Dicho lo anterior y teniendo en cuenta la sobrecarga que conllevaría representar cada concepto del retículo en la Figura 2.2, se puede mencionar aquí como ejemplo, el concepto etiquetado como 'Oriente' *(Oriente Medio)*, el cual tiene como extensión el conjunto {*Singapore Airlines, The Austrian Airlines Group, Lufthansa, Air Canada*}, y como intensión {*Oriente Medio, Canadá, Estados Unidos, Europa, Asia*}.

En la parte superior del retículo, encontramos los destinos que ofrecen

la mayoría de las aerolíneas: Europa, Asia Pacífico y los Estados Unidos. Por ejemplo, exceptuando British Midland y Ansett Australia, todas las aerolíneas ofrecen vuelos a Estados Unidos. Esas dos líneas aéreas se encuentran en la parte superior del diagrama, ya que tienen la menor oferta de vuelos, operan sólo en Europa y Asia, respectivamente.

De esta forma, cuanto más se descienda en el retículo de conceptos, más destinos ofrecerán las aerolíneas. Así, la mayor oferta de destinos se encuentra en las aerolíneas de la parte inferior del retículo: Lufthansa y Air Canada. De forma análoga, conforme más se descienda en el retículo, se encontarán los destinos menos ofertados, e.g. África, Oriente Medio y el Caribe.

2.1.5. Sistema de Implicaciones

Una noción equivalente al retículo de conceptos es el denominado sistema de implicaciones [46]. Como ya se adelantó en el Capítulo 1, las implicaciones constituyen el eje fundamental de trabajo en esta tesis doctoral.

Para introducir la labor realizada utilizando la lógica sobre los conjuntos de implicaciones, se va a comenzar describiendo los componentes habituales de una lógica: su lenguaje, semántica, sistema axiomático y método de razonamiento automático. No obstante, como se ha comentado anteriormente, los conjuntos de implicaciones se han utilizado para diferentes campos de conocimiento (FCA, bases de datos). En consecuencia, en este punto se presenta el elemento de la lógica común a todos ellos, en concreto, el lenguaje. El resto se presentará para cada una de las secciones correspondientes 2.2 y 2.3.

Lenguaje

Definición 2.1.8. *Dado un conjunto M finito de símbolos (denominados atributos) no vacío, el lenguaje sobre M se define como:*

$$\mathcal{L}_M = \{A \to B \mid A, B \subseteq M\}$$

Las fórmulas $A \to B$ se denominan *implicaciones* y los conjuntos A y B reciben el nombre de *premisa* y *conclusión* de la implicación respectivamente. Los conjuntos $\sum \subseteq \mathcal{L}_M$ se denominan *conjuntos de implicaciones* sobre M.

Se utilizará la siguiente notación:

- Se utilizarán letras minúsculas para denotar los elementos en M, mientras que las mayúsculas denotan sus subconjuntos.
- Se omiten las llaves en premisas y conclusiones, es decir, *abcde* denota el conjunto $\{a, b, c, d, e\}$.
- Se escriben los elementos de 2^M por yuxtaposición, es decir, para $X \cup Y$ se escribirá XY.
- Para la diferencia se utiliza $X \setminus Y$.

Semántica

Tras la definición del lenguaje se pasa ahora a introducir la semántica utilizada, para lo cual se va a utilizar la noción de *operador de cierre* definido anteriormente.

Definición 2.1.9 (Modelo)**.** *Sea $K = (G, M, I)$ un contexto formal y sea $A \to B \in \mathcal{L}_M$. El contexto K es un modelo para $A \to B$ si $B \subseteq A''$. Se denota por $K \models A \to B$.*

La noción de modelo puede extenderse a los sistemas de implicaciones de la siguiente manera. Dado $\Sigma \subseteq \mathcal{L}_M$, la expresión $K \models \Sigma$ indica que $\Sigma \models A \to B$ para toda $A \to B \in \Sigma$.

Ejemplo 2.1.10. *De la información mostrada en el contexto formal del Ejemplo 2.1.2 se puede identificar que: $K \models Caribe \to Latinoamerica$. Y también que: $K \not\models Caribe \to Mexico$.*

Al comenzar el capítulo se habló de la equivalencia entre la información que muestra el retículo de conceptos y las implicaciones; tras haber

introducido formalmente el lenguaje y la semántica de las implicaciones, ahora ya se pueden mencionar algunos ejemplos que reflejen esta equivalencia utilizando la información del retículo de la Figura 2.2. Así, es posible identificar ejemplos de implicaciones tales como:

- $Canada \rightarrow Estados\ Unidos$, ya que se puede ver como el primer atributo aparece por primera vez en la jerarquía en un subconcepto del nodo que refleja la primera aparición del segundo. En otras palabras, todo objeto g del contexto que tenga el atributo $Canada$ tendrá el atributo $Estados\ Unidos$.

Se puede considerar a las implicaciones y el retículo de conceptos como distintas alternativas para tratar la información que se puede extraer de un contexto formal. No obstante, los retículos de conceptos permiten una representación gráfica, mientras que la diferencia fundamental, desde el punto de vista de esta tesis doctoral, es que los sistemas de implicaciones proporcionan una manipulación simbólica usando la lógica de implicaciones, y por tanto, permiten el razonamiento automático.

Como se ha mencionado anteriormente, se entra ahora en la segunda sección de este capítulo en la cual se van a utilizar las implicaciones en el ámbito de las bases de datos relacionales, donde recibirán el nombre de Dependencias Funcionales.

2.2. Bases de Datos Relacionales

El modelo de base de datos relacional aparece en el reconocido artículo de Edgar Frank Codd en 1970 [21]. Codd propuso que los sistemas de bases de datos deberían presentarse al usuario mediante una vista de datos organizada en forma de tablas bidimensionales (filas y columnas) que denominó *relaciones*.

Formalmente, en el modelo relacional, una base de datos consiste en una o más relaciones [38]. Una relación R definida sobre un conjunto de dominios D_1, D_2, ..., D_n está formada por un esquema y un cuerpo como se definen a continuación.

Definición 2.2.1 (Esquema)**.** *Un esquema E para una relación R se define como un conjunto fijo de pares (atributo:dominio). Habitualmente se denota como $\{(A_1 : D_1),\ (A_2 : D_2),\ (A_n : D_n)\}$ donde cada A_j corresponde a un único D_j y los A_j son todos distintos.*

En el Ejemplo 1.1.1, el esquema sería:

{

Título:Cadena de caracteres, Año:Número natural, País:Cadena de caracteres, Director:Cadena de caracteres, Nacionalidad:Cadena de caracteres, Actor:Cadena de caracteres

}.

El conjunto de los esquemas para las relaciones de una base de datos se denomina *esquema de base de datos relacional*.

Definición 2.2.2 (Cuerpo)**.** *Un cuerpo C para una relación R se define como un conjunto de tuplas de pares (atributo:valor). Habitualmente se denota como $\{(A_1 : v_{i1}),\ (A_2 : v_{i2}),\ (A_n : v_{in})\}$ con $i = 1, 2, ..., m$ tal que m es el cardinal de R, i.e. el número de tuplas de la relación. En cada $(A_j : v_{ij})$ se tiene que $v_{ij} \in D_j$.*

En el Ejemplo 1.1.1, un extracto del cuerpo sería:

{

[(Título:Pulp Fiction), (Año:1994), (País:USA), (Director:Quentin Tarantino), (Nacionalidad:USA), (Actor:Uma Thurman)],

[(Título:King Kong), (Año:2005), (País:NZ), (Director:Peter Jackson), (Nacionalidad:NZ), (Actor:Naomi Watts)],

...

}.

Tras haber definido formalmente el cuerpo y el esquema de un relación, se puede mencionar que en los sistemas gestores de bases de datos relacionales, las relaciones se representan como una tabla de doble entrada, en la que (con un una terminología más informal) se encuentran los siguientes elementos:

- **Atributo**. Cada una de las columnas de una relación se identifica con un atributo.

- **Tupla**. Se identifica cada tupla de una relación con una fila de la tabla, exceptuando la fila que contiene a los atributos.

2.2.1. Dependencias Funcionales

Un diseño defectuoso del esquema de base de datos relacional puede provocar lo que se denominan *anomalías* [61, 108, 120]. Estas anomalías son problemas que aparecen a la hora de operar con la base de datos en términos de actualizaciones, eliminaciones o inserción de información redundante [30, 121].

La manera de eliminar estas anomalías consiste en descomponer las relaciones [5]. La descomposición de relaciones conduce al campo de la *normalización* [23] que si bien no forma parte del ámbito principal de esta tesis doctoral, permite dar paso a la introducción del concepto de DF, el cual nace como un recurso para caracterizar la semántica de las relaciones y, por consiguiente, permitir hacer una aplicación de las transformaciones de normalización [32]. Ésta es una de las diferencias del tratamiento de las relaciones entre FCA y bases de datos relacionales, ya que para el primero, lo que se pretende es capturar conocimiento, mientras que en el segundo, las DFs se utilizan para definir formas normales [5, 41, 64].

Teniendo en cuenta que la sintaxis que se va a utilizar para DFs es la misma que la ya introducida para las implicaciones en FCA en la Sección 2.1.5, donde el conjunto M lo forman los atributos del esquema de la relación, se pasa ahora a describir la semántica para las DFs.

Semántica

Definición 2.2.3 (Dependencia Funcional)**.** *Una dependencia funcional $X \to Y$ se cumple en una tabla R si y sólo si para cada dos tuplas de R, si sus valores en X coinciden, entonces también coinciden sus valores en Y.*

Intuitivamente, una DF $X \to Y$ indica que el conjunto de atributos X determina el conjunto de atributos Y, lo cual refleja fielmente la noción de función (y de ahí su nombre) entre los dominios de A y B, $f : A \to B$. Estas funciones son las que se usan para tratar las anomalías mencionadas anteriormente, buscando aquellas funciones que se correspondan con relaciones de forma unívoca. El siguiente ejemplo ilustra cómo la semántica de la DFs y las implicaciones en FCA es diferente.

Ejemplo 2.2.4. *Sea el contexto formal $K = (G, M, I)$ representado a continuación:*

	a	b	c
g_1	*1*	*0*	*1*
g_2	*1*	*1*	*1*
g_3	*0*	*0*	*0*
g_4	*0*	*1*	*1*

Dado ese contexto formal, se puede ver que se verifica la implicación $a \to c$ en K, mientras que por otro lado, al considerar el contexto formal como una relación del modelo relacional y considerando la semántica de DF, no se cumple que $a \to c$ debido a los diferentes valores de g_3 y g_4 en el atributo c.

La forma en que las DFs se asemejan a las implicaciones no viene dada porque se parezcan en su semántica sino por el hecho de que se pueden manejar utilizando la misma lógica, los axiomas de Armstrong [4], que se presentarán más adelante en la Sección 2.3.1.

La diferencia principal en la interpretación de las DFs y las implicaciones proviene de la estructura de los *datasets* considerados, que en el caso de FCA son binarios mientras que en el modelo relacional son multivaluados. La noción de implicaciones en FCA puede ampliarse para poder tratar con contextos formales donde la relación I toma valores en un conjunto en lugar de ser binaria. Este tipo de contextos formales se denominan multivaluados y su aproximación se realiza a través del llamado análisis de conceptos formales difusos [8, 18], lo que queda fuera del ámbito de esta tesis doctoral.

2.3. Lógica de Implicaciones

En esta sección se van a introducir las lógicas de implicaciones que permitirán disponer de sistemas axiomáticos correctos y completos para trabajar con ellas. Para ello, el punto de partida van a ser los axiomas de Armstrong [4] como pionero en el campo. Sin embargo, el sistema de Armstrong no es adecuado para el razonamiento automático debido a su fuerte dependencia a la transitividad. En consecuencia, tras los Axiomas de Armstrong se presenta la lógica SL_{FD}, la cual sí va a permitir desarrollar métodos de razonamiento automático.

Ambas lógicas pueden aplicarse con éxito para trabajar con implicaciones y con DFs. El lenguaje sobre el que van a trabajar ambas lógicas es el mismo (ver Definición 2.1.8). Del mismo modo, la semántica puede verse en sendas definiciones, 2.1.9, 2.2.3, para implicaciones y DFs respectivamente. A continuación, se analizarán los sistemas axiomáticos y métodos de razonamiento. Para los siguientes apartados, y por facilitar la redacción y lectura, se utiliza el término implicaciones, siendo igualmente válido para DFs.

2.3.1. Axiomas de Armstrong

Los denominados axiomas de Armstrong son el primer sistema axiomático descrito para tratar sistemas de implicaciones utilizando la lógica [4]. Este sistema ha tenido una clara influencia en el diseño de varias lógicas sobre implicaciones, todas ellas construidas alrededor del paradigma de la transitividad [6, 58, 93].

Este sistema axiomático está formado por un esquema de axioma, denominado *Reflexivo*:

$$\texttt{[Ref]}\ \frac{}{AB \to A}$$

y dos reglas de inferencia, *Aumentativa* y *Transitiva* que se definen como:

$$\texttt{[Aug]}\ \frac{A \to B}{AC \to BC} \qquad \texttt{[Tran]}\ \frac{A \to B,\ B \to C}{A \to C}$$

Para el sistema axiomático anterior, la noción de *deducción* ($\vdash$) se define como:

Definición 2.3.1. *Se dice que una implicación $A \to B$ se deriva sintácticamente (o se deduce) de un sistema de implicaciones Σ, y se denota por $\Sigma \vdash A \to B$, si existe una secuencia de implicaciones $\sigma_1, \ldots, \sigma_n \in \mathcal{L}_M$ tal que $\sigma_n = A \to B$ y, para todo $1 \leq i \leq n$, la implicación σ_i satisface una de las siguientes condiciones:*

- *σ_i es un axioma, es decir, verifica el esquema* `[Ref]`.
- *$\sigma_i \in \Sigma$.*
- *σ_i se obtiene a partir de implicaciones pertenecientes a $\{\sigma_j \mid 1 \leq j < i\}$ aplicando las reglas de inferencia del sistema axiomático.*

La secuencia $\sigma_1, \ldots, \sigma_n$ constituye una demostración para $\Sigma \vdash A \to B$

Los axiomas de Armstrong son un sistema axiomático correcto y completo tanto si se está trabajando con implicaciones en FCA [4] como con DFs en bases de datos relacionales [46].

Debido al rol central que desempeña la transitividad en ese sistema axiomático, el desarrollo de métodos ejecutables para resolver problemas de implicaciones se ha mostrado infructuoso y se ha tenido que recurrir a métodos indirectos como se verá en la Sección 2.4. No obstante, como se ha mencionado, es un buen punto de partida para el desarrollo de nuevas lógicas para el tratamiento de las implicaciones, en concreto, para la lógica SL_{FD}, que se introduce a continuación.

2.3.2. Lógica de Simplificación

La lógica SL_{FD} no toma el paradigma de la transitividad como centro sino que se guía por la idea de simplificar el conjunto de implicaciones mediante la eliminación de atributos redundantes de manera eficiente [26]. Por consiguiente, la introducción de la lógica SL_{FD} abrió la puerta al desarrollo de métodos de razonamiento automatizados directamente basados en su novedoso sistema axiomático [27, 29]. La posibilidad de desarrollar estas

aplicaciones ha sido la motivación principal para el desarrollo de esta tesis, buscando sobre todo, mejorar la eficacia y la eficiencia.

Seguidamente, se va a comenzar definiendo su sistema axiomático que, como ya se ha comentado, es válido para ambas semánticas (FCA, DFs) y además es común debido a que utilizan el mismo lenguaje.

Sistema Axiomático

SL_{FD} se define como el par $(\mathcal{L}_M, S_{FD})$ donde S_{FD} tiene el siguiente esquema de axioma:

$$\texttt{[Ref]} \quad \frac{}{AB \to A}$$

junto con las siguientes reglas de inferencia, denominadas *fragmentación*, *composición* y *simplificación* respectivamente.

$$\texttt{[Frag]} \ \frac{A \to BC}{A \to B} \qquad \texttt{[Comp]} \ \frac{A \to B,\ C \to D}{AC \to BD}$$

$$\texttt{[Simp]} \ \text{Si } A \subseteq C, A \cap B = \varnothing, \ \frac{A \to B,\ C \to D}{A(C \smallsetminus B) \to D}$$

SL_{FD}, al igual que los axiomas de Armstrong, constituye una lógica correcta y completa tanto para implicaciones como DFs, tal y como los autores presentaron en [26] y como demuestra el siguiente teorema.

Teorema 2.3.2. *Sea M un conjunto finito no vacío de atributos, $\Sigma \subseteq \mathcal{L}_M$ y $A \to B \in \mathcal{L}_M$. Entonces, $\Sigma \models A \to B$ si y sólo si $\Sigma \vdash A \to B$.*

Se destaca que el lenguaje SL_{FD} considera como fórmulas válidas aquellas en donde cualquiera de sus dos partes puede ser el conjunto vacío, es decir, de los tipos $A \to \varnothing$ y $\varnothing \to A$, tal y como los autores discutieron en [26].

La principal ventaja de la lógica SL_{FD} radica en que las reglas de inferencia pueden considerarse reglas de equivalencia. Como consecuencia, se ha podido utilizar como núcleo principal para el desarrollo de métodos automáticos para diversas aplicaciones (e.g. obtener claves minimales,

cálculos del cierre) como se verá más adelante. Un estudio más detallado al respecto, incluyendo teoremas y sus demostraciones, puede verse en [87].

Para finalizar, estos sistemas axiomáticos se pueden utilizar para el desarrollo de métodos de razonamiento automático con los que resolver el denominado problema de la implicación tal y como se muestra en el siguiente apartado.

2.4. Razonamiento Automático

Una vez introducido el lenguaje de la lógica, la semántica (en las dos acepciones que se han utilizado en esta tesis) y el sistema axiomático, se llega ahora a la última sección de estos preliminares. En ella se va a abordar la demostración automática a través del problema de la implicación y haciendo uso del cierre de atributos sobre el sistema axiomático.

Básicamente, el problema de la implicación es el siguiente: dado un conjunto de implicaciones Σ, determinar mediante la lógica si a partir de ese conjunto se puede inferir la validez de una nueva implicación dada $\Sigma \vdash A \rightarrow B$ [7, 76, 85].

A continuación se introduce el cierre de atributos, que se utiliza como base para construir el demostrador automático.

Definición 2.4.1. *Dado un conjunto $X \subseteq M$, llamamos cierre de X sobre Σ (notado X_Σ^+) como el mayor subconjunto de M tal que $\Sigma \vdash X \rightarrow X_\Sigma^+$.*

La definición anterior de cierre sintáctico de atributos da lugar a un operador de cierre, siempre y cuando el conjunto Σ sea un sistema de implicaciones completo (las implicaciones deducibles del contexto formal son todas inferibles desde Σ). No se entra en detalle sobre estos aspectos formales en estos preliminares puesto que, en todo el trabajo, se parte de la noción de cierre sintáctico como base.

En [87] los autores presentan un novedoso método para calcular el cierre de atributos sobre un conjunto de implicaciones, basado en la lógica SL_{FD}.

Se pasa ahora a presentar los métodos de razonamiento utilizados en esta tesis doctoral: el cierre clásico de Maier [75] como trabajo seminal

de este área, y a continuación, el cierre SL_{FD}, este último como núcleo principal del funcionamiento de los métodos diseñados y utilizados en las publicaciones que avalan este trabajo de tesis [13–15].

Para abordar el problema de la implicación, se utiliza el cierre sintáctico a modo de demostrador automático, pues dado un conjunto de implicaciones Σ, debido a la definición de cerrado anterior, se tiene que $\Sigma \vdash A \to B$ si y sólo si $B \subseteq A^+_\Sigma$. Este resultado hace que la definición de un método para calcular el cierre sintáctico abra la puerta al tratamiento automático del problema de la implicación. Ello será el centro del siguiente apartado.

2.4.1. Algoritmos para el Cálculo del Cierre

El problema de la implicación se ha abordado tradicionalmente utilizando un método básico que recibe como entrada un conjunto de atributos $X \subseteq M$ y un conjunto de implicaciones $\Sigma \subseteq \mathcal{L}_M$ y utiliza de forma exhaustiva la relación de subconjunto recorriendo iterativamente Σ y agregando nuevos elementos al cierre. Este método, fue propuesto en la década de 1970 por Maier [75] y puede considerarse la base principal donde se han sustentado tantos otros. Su funcionamiento puede verse en detalle en el Algoritmo 2.1.

Algoritmo 2.1: Cierre clásico

Entrada: Σ, A
Salida: A^+_Σ

```
1  inicio
2      A+Σ := A
3      repetir
4          A' := A+Σ
5          para cada X → Y ∈ Σ hacer
6              si (X ⊆ A+Σ) y (Y ⊈ A+Σ) entonces
7                  A+Σ := A+Σ ∪ {Y}
8      hasta A+Σ = A'
9      devolver A+Σ
```

En [93], los autores muestran que la complejidad del problema del cierre es $\mathtt{O}(|A|\,|\Sigma|)$. En [87] se presentó el método del cierre basado en la lógica SL_{FD}. En dicho trabajo además se muestra que dicho método presenta un mejor rendimiento que los métodos anteriores. Como novedad, la salida del nuevo método de cierre no es sólo el conjunto cerrado sino que también produce un conjunto de implicaciones que puede ser interpretado como el conocimiento que resta en el sistema y que complementa a los atributos que están en el cerrado.

El método de razonamiento automático en SL_{FD} se basa en el Teorema de la deducción y un conjunto de equivalencias. Introduciremos ambas cosas antes de presentar el método.

Siguiendo la forma habitual, dos sistemas de implicaciones $\Sigma_1, \Sigma_2 \subseteq \mathcal{L}_M$ se dicen equivalentes si para toda implicación $A \to B$ de Σ_1, se tiene que $\Sigma_2 \vdash A \to B$ y viceversa.

Teorema 2.4.2 (Teorema de la deducción)**.** *Sea $A \to B \in \mathcal{L}_M$ y $\Sigma \subseteq \mathcal{L}_M$. Entonces,*

$$\Sigma \vdash A \to B \quad \textit{si y sólo si} \quad \{\varnothing \to A\} \cup \Sigma \vdash \{\varnothing \to B\}$$

La siguiente proposición proporciona tres equivalencias, denominadas también: *Fragmentación, Composición y Simplificación.*

Proposición 2.4.3. *Sean $A, B, C, D \subseteq M$. Se verifican las siguientes equivalencias:*

(I) $\{A \to B\} \equiv \{A \to B \setminus A\}$

(II) $\{A \to B, A \to C\} \equiv \{A \to B \cup C\}$

(III) $\{A \to B, C \to D\} \equiv \{A \to B, C \setminus B \to D \setminus B\}$ *siendo $A \cap B = \varnothing$ y $A \subseteq C$*

Mediante el Teorema de la deducción y las equivalencias anteriores, se puede pasar de las reglas de inferencia del sistema axiomático clásico de Armstrong a un sistema que puede ser automatizable. En este punto aparece el cierre SL_{FD}, que los autores denominaron `Cls` [87], y que actúa según el siguiente procedimiento.

Los pasos del algoritmo desglosados en lenguaje natural y de forma más detallada son:

(I) En primer lugar, se incluye la fórmula $\varnothing \to A$ en Σ y se usa como semilla por el método de razonamiento mediante las equivalencias mencionadas en la proposición anterior.

(II) A continuación, el algoritmo entra en un proceso iterativo en el cual se irán aplicando las siguientes equivalencias hasta alcanzar un punto en el que no sea posible aplicar ninguna.

- **Eq. I**: Si $B \subseteq A$ entonces $\{\emptyset \to A, B \to C\} \equiv \{\emptyset \to A \cup C\}$.
- **Eq. II**: Si $C \subseteq A$ entonces $\{\emptyset \to A, B \to C\} \equiv \{\emptyset \to A\}$.
- **Eq. III**: En otro caso $\{\emptyset \to A, B \to C\} \equiv \{\emptyset \to A, B \smallsetminus A \to C \smallsetminus A\}$.

(III) En el momento en el que no sea posible aplicar ninguna de las equivalencias anteriores, el algoritmo termina.

Formalmente, el algoritmo `Cls` puede verse detalladamente en el pseudocódigo 2.2.

Aunque es cierto que existen en la literatura numerosas propuestas de algoritmos para calcular el cierre (la mayoría de ellas como modificaciones del cierre clásico de Maier [75]), la principal novedad y ventaja que aporta `Cls` es que, aparte del cálculo del cierre, y de manera simultánea, también se calcula el conjunto reducido de implicaciones, guardando de esta forma el conocimiento complementario que describe la información que no pertenece al cierre. Este hecho conduce sin lugar a dudas a una posición privilegiada, ya que evita el elevado coste de minería de datos para extraer el nuevo conjunto de implicaciones para el *dataset* reducido después de cada aplicación del método, algo imprescindible con las implementaciones clásicas. Esta característica del algoritmo es la que se ha explotado ampliamente en los resultados obtenidos a lo largo de esta tesis doctoral.

Para finalizar este apartado, se muestra el siguiente ejemplo básico de aplicación del algoritmo `Cls` propuesto.

Algoritmo 2.2: Cierre `Cls`

Entrada: Σ, A
Salida: A^+_Σ, Σ'

```
1  inicio
2      A⁺_Σ := A
3      Σ' := Σ
4      repetir
5          A' := A⁺_Σ
6          para cada  B → C ∈ Σ hacer
7              si B ⊆ A⁺_Σ entonces
8                  (Eq. I)
9                  A⁺_Σ := A⁺_Σ ∪ {B}
10                 Σ' := Σ' ∖ B → C
11             si no si C ⊆ A⁺_Σ entonces
12                 (Eq. II)
13                 Σ' := Σ' ∖ B → C
14             si no si (B ∩ A⁺_Σ ≠ ∅) o (C ∩ A⁺_Σ ≠ ∅) entonces
15                 (Eq. III)
16                 Σ' := Σ' ∪ {B ∖ A⁺_Σ → C ∖ A⁺_Σ}
17     hasta A⁺_Σ = A'
18     devolver A⁺_Σ, Σ'
```

Ejemplo 2.4.4. *Sean* $\Sigma = \{ak \to bc, cd \to gh, cij \to kl, de \to f, g \to de, hf \to ia, f \to c\}$ *y* $A = \{adf\}$.

La siguiente tabla muestra la traza de ejecución paso a paso del algoritmo.

				Iteración 0			
Σ'	$ak \to bc$	$cd \to gh$	$cij \to kl$	$de \to f$	$g \to de$	$hf \to ia$	$f \to c$
A^+_Σ	adf						
				Iteración 1			
Σ'	$k \to bc$	$c \to gh$	$cij \to kl$	$\times$	$g \to e$	$h \to i$	$\times$
Eq	III	III	$-$	II	III	III	I
A^+_Σ							$acdf$
				Iteración 2			
Σ'	$k \to b$	$\times$	$ij \to kl$		$\times$	$\times$	
Eq	III	I	III		I	I	
A^+_Σ		$acdfgh$			$acdefgh$	$acdefghi$	
				Iteración 3			
Σ'	$k \to b$		$j \to kl$				
Eq	$-$		$-$				
A^+_Σ	$\boldsymbol{acdefghi}$						

El resultado del algoritmo `Cls` *es:*

- *En primer lugar, el cierre* $A^+_\Sigma = \{a, c, d, e, f, g, h, i\}$.
- *Y en segundo lugar y muy importante tal y como se ha mencionado, el nuevo conjunto reducido de implicaciones* $\Sigma' = \{k \to b, j \to kl\}$, *que contiene la información que no pertenece al cierre.*

Capítulo 3

Claves Minimales

Recuerde, mi amigo, que el conocimiento es más fuerte que la memoria, y no debemos confiar en lo más débil.

Drácula

B. Stoker

Título:	Reducing the search space by closure and simplification paradigms. A parallel key finding method
Autores:	Fernando Benito-Picazo, Pablo Cordero, Manuel Enciso, Ángel Mora
Revista:	The Journal of Supercomputing, Springer
Factor Impacto JCR:	1,349. Posición 52 de 104 (Q2)
Año:	2016
Categoría:	Computer Science, Theory & Methods
Publicación:	14 enero 2016
DOI:	10.1007/s11227-016-1622-1

J Supercomput (2017) 73:75–87
DOI 10.1007/s11227-016-1622-1

Reducing the search space by closure and simplification paradigms

A parallel key finding method

F. Benito-Picazo[1] · P. Cordero[1] · M. Enciso[1] · A. Mora[1]

Published online: 14 January 2016

Abstract In this paper, we present an innovative method to solve the minimal keys problem strongly based on the Simplification Logic for Functional Dependencies. This novel method improves previous logic-based methods by reducing, in a significant degree, the size of the search space this problem deals with. Furthermore, the new method has been designed to easily fit within a parallel implementation, thereby increasing the boundaries current methods can reach.

Keywords Minimal keys · Parallelism · Logic

1 Introduction

Finding a set of items/attributes characterizing a greater set of data is a significant problem in several areas: data modeling, data integration, integration of heterogeneous databases, knowledge discovery, anomaly detection, query formulation, query optimization, and indexing. This problem is also interesting in emergent areas as

✉ F. Benito-Picazo
fbenito@lcc.uma.es

P. Cordero
pcordero@uma.es

M. Enciso
enciso@uma.es

A. Mora
amora@ctima.uma.es

[1] University of Málaga, Andalucía Tech, Málaga, Spain

linked data. In [14] authors delimit the problem: "establishing semantic links between data items can be really useful, since it allows crawlers, browsers and applications to combine information from different sources". The proposal is the use of keys to infer identity links among the data distributed in the web.

The notion of key is fundamental in Codd's relational model of data [2]. Calculating all minimal keys constitutes a tough problem; in [10,22], authors prove that the number of minimal keys for a relational system can be exponential in the number of attributes, or factorial in the number of dependencies. The main approaches to this problem point to the works of Lucchesi and Osborn in [10] that show an algorithm to calculate all candidate keys. Saiedian and Spencer [15] presented an algorithm using attribute graphs to find all possible keys of a relational database schema, and Wastl [20] proposed an inference system for deriving all keys which can be inferred from a set of functional dependencies. Recent works about calculating minimal keys are [16,21] and a very modern paper approaching the problem in a logic style [5]. In addition, in [9,18, 19] the authors proposed the use of formal concept analysis [6] to face problems related with the discovery and management of implications, which can be considered complementary to our work.

We highlight [20] to be the one closest to our framework. In that work, a tableaux method to calculate all minimal keys is presented. In [3] we introduce the Key Algorithm, a method inspired in Wastl's tableaux method where the Simplification Logic for Functional Dependencies [12] is used to find all minimal keys. Later, in [5] we introduce the SST method, achieving a great performance in comparison to its predecessors based on the introduction of the inclusion-minimality test which avoids the opening of extra-branches of the tree, and the search space becomes narrower.

A parallel algorithm for the manipulation of implications is described in [17], enclosed in the field of hyper-graphs. Parallel issues in similar problems have been addressed by several authors. For instance, Krajca et al. [8] present a parallel algorithm for computing formal concepts. A first view for the parallelization of Wastl's method and the Key Algorithm was presented in [1], showing how parallelism could naturally be integrated in tableaux methods.

This paper makes the following contributions. We propose a new method named Closure Keys incorporating an efficient prune mechanism that use the closure method based on $\mathbf{SL}_{\mathrm{FD}}$ to improve SST method. The new method is based on the strong relation between the notion of key and the closure operator, not only in the definition issue but also in its construction. The closure operator defined in [12] allows us to highly reduce the search space by introducing shortcuts in the way to the leaves, where keys are finally produced. Moreover, a parallel implementation of SST method is presented along with several experiments to confirm the improvements that have been accomplished.

The remainder of the paper is organized as follows: Sect. 2 presents the key finding problem and the state of the art. In Sect. 3, we explain how a closure operator could help us improving the efficiency over minimal keys methods. In Sect. 4, the main aspects of the implementation of these methods will be exposed. Several results are shown in Sect. 5 with a collection of tables and charts. Finally, conclusions will end up the document.

2 Background

In this section we introduce the minimal key problem, the basic notions related with this issue and the last logic-based algorithm in the literature to tackle this problem.

Definition 1 (*Functional dependency*) Let Ω be a set of attributes. A functional dependency (FD) over Ω is an expression of the form $X \to Y$, where $X, Y \subseteq \Omega$. It is satisfied in a table R if for every two tuples of R, if they agree on X, then they agree on Y.

A key of a relational table is an attribute subset that allows us to uniquely characterize each row, and it is defined by means of FDs as follows:

Definition 2 (*Key*) Given a table R over the set of attributes Ω, we say that K is a key in R if the functional dependency $K \to \Omega$ holds in R.

We would like to manage keys with no superfluous attributes, named minimal keys. Due to space limitation, we refer those readers non-familiar with the formal notions of FDs, keys and relational tables to [11]. In addition, it is remarkable that this classical problem appears in several areas. For instance, in [5], we show how the minimal key problem in databases has an analogous one in formal concept analysis, where the role of FDs is played by attribute implications. In that paper, minimal key problem was presented from a logical point of view. An axiomatic system to manage both FDs and implications, named Simplification Logic and denoted $\mathbf{SL}_{\mathrm{FD}}$, was introduced in [4]. The inference system of such logic is presented as follows:

Definition 3 (*Axiomatic system*) $\mathbf{SL}_{\mathrm{FD}}$ has a reflexivity axiom `[Ref]` $\frac{B \subseteq A}{A \to B}$; and inference rules named fragmentation, composition and simplification.

$$\texttt{[Frag]}\ \frac{A \to BC}{A \to B}; \qquad \texttt{[Comp]}\ \frac{A \to B,\ C \to D}{AC \to BD}; \qquad \texttt{[Simp]}\ \text{If } A \subseteq C, A \cap B = \varnothing,\ \frac{A \to B,\ C \to D}{C\text{-}B \to D\text{-}B}$$

This logic allows the development of efficient methods to manage FDs and implications. In [5] a new algorithm, named SST, for computing all minimal keys using a tableaux-like strategy was introduced, opening the door to embed a massive parallelism into its execution.

SST relies on the notion of set closure, a basic notion in database theory which allows to characterize the maximum attribute set that can be reached from a given attribute set A with respect to a set of FDs using the axiomatic system. Thus, if the closure of A is denoted as A^+_Γ, the inference system for FDs allows us to infer the FD $A \to A^+_\Gamma$. The logical-style approach to the minimal key problem consists in the enumeration of all attribute sets A such that the following FD holds $A \to \Omega$.

SST algorithm is strongly based on two proper extensions of the $\mathbf{SL}_{\mathrm{FD}}$ simplification rule, named `[sSimp]` and `[lSimp]`. These rules guide the construction of the search space to look for all minimal keys (see [5] for further details). The method works step by step by building a tree from the original problem to the solution, i.e. the set of minimal keys. They are applied to each pair (attribute set, implication set) labeling each node to produce new subnodes that will be the origin of new branches in the search space. Applying `[lSimp]` to the subset of attributes in each node and the implication

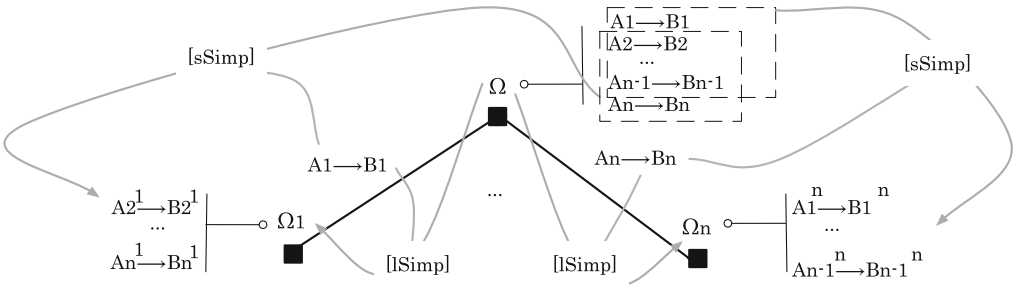

Fig. 1 `[lSimp]` generates new implication subsets and `[lSimp]` generates new roots

in the corresponding edge, we obtain the new root in the branch (e.g. in Fig. 1 from Ω and $A_1 \rightarrow B_1$, we obtain the new subset Ω_1). Moreover, the application of `[sSimp]` over the implication set in each node is done by taking every implication as a pivot and applying the rule to the rest of the implications with the corresponding pivot to generate new branches (e.g. in Fig. 1 the first branch is generated by taking $A_1 \rightarrow B_1$ as the pivot and facing it to the rest of the implication set $A_2 \rightarrow B_2, \ldots, A_n \rightarrow B_n$).

When the set of implications in a node is the empty set, we have reached a leaf of the tree and a minimal key is added to the solution. The full situation is depicted in Fig. 1, where a schema of the application of both rules to a generic node is shown.

SST shows a great performance compared with its predecessors. The main benefit in the reduction of the search space was the introduction of the inclusion-minimality test to avoid the opening of extra-branches. Thus, SST does not open some branches which are going to produce the same keys that are calculated in another branch. The characterization of such branches rendering leaves with duplicate information is not a trivial issue. To approach it, we have defined the notion of minimal implication w.r.t. an implication set Γ: $A \rightarrow B \in \Gamma$ is minimal if for all $C \rightarrow D \in \Gamma$ we have that $C \not\subset A$.

3 Improving minimal keys methods by means of closure operator

As it has been established this far, the enumeration of all minimal keys is a non-trivial problem that can be approached using logic-based methods. The main goal to progress in this line is the reduction of the search space that can be measured by means of the number of nodes in the tree. In the previous section, we have summarized the main characteristics of the SST method and how it incorporates a strategy to shorten the width of the tree corresponding to the search space. Thus, it provides a narrower tree regarding previous methods. In this spirit, we have studied how to reduce even more the size of the tree by shorting its depth. The kernel of the method presented here is a logic-based closure algorithm presented in [12], strongly based on Simplification Logic $\mathbf{SL}_{\mathrm{FD}}$. The closure of an attribute set can be solved in linear time w.r.t. the cardinality of the implication set. There exists in the literature several approaches to tackle this problem, but most of them may be considered a modification of classical Maier's method [11].

The method presented in [12], named $\mathbf{SL}_{\text{FD}}$ Closure, can be considered a novel approach to classical closure methods, because it provides a new feature: In addition to the set of attributes that constitutes the output of the closure operator, $\mathbf{SL}_{\text{FD}}$Closure provides a subset of implications of the original Γ set. This new implication subset encloses the knowledge that can be considered as the complement of the closure and, as we shall see, may be used in a very smart way to find minimal keys by means of the closure operator.

As mentioned before, closure operator may be used to define the notion of key. In this section we propose it as the backbone of a new method to compute all minimal keys. In this approach, $\mathbf{SL}_{\text{FD}}$ Closure plays an outstanding role. The main point of our method is that it receives a set of implications Γ and a subset of attributes $X \subseteq \Omega$. It renders the closure set X^+ w.r.t. Γ, and also, a novelty output Γ' consisting in a new implication set that covers the semantics that reaches beyond X^+. If $\Gamma' = \varnothing$ then $X^+ = \Omega$ (see [12] for further details).

We propose to apply $\mathbf{SL}_{\text{FD}}$ Closure to each minimal implication in the Γ set in each node to open new branches with the output produced by $\mathbf{SL}_{\text{FD}}$ Closure. The definition of our new minimal key method is presented in Algorithm 1. An illustrative example showing the reduction in the tree provided by Closure Keys w.r.t. SST method is shown in Figs. 2 and 3 respectively.

The main call of Algorithm 1 must be Closure_Keys(Ω, Γ, $\varnothing$, $\varnothing$). Later, to get all minimal keys we pick up the minimal elements of the fourth parameter (which acts as an accumulator in a in/out mode) of this procedure call with regard to $(2^{\Omega}, \subseteq)$. The following theorem ensures that Algorithm 1 provides a sound and complete method.

Theorem 1 *Let Ω be an attribute set and Γ be an implicational system over M. Then Algorithm 1 renders all minimal keys.*

Proof Firstly, the termination of the algorithm is ensured because, in each recursive call of Closure_Keys, the cardinality of the implicational system is strictly decreased.

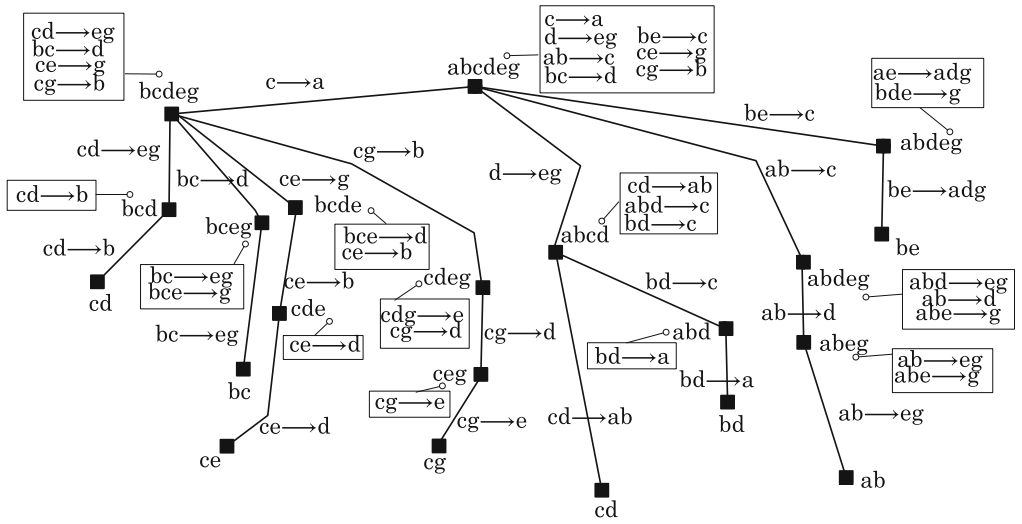

Fig. 2 A complete example using SST method

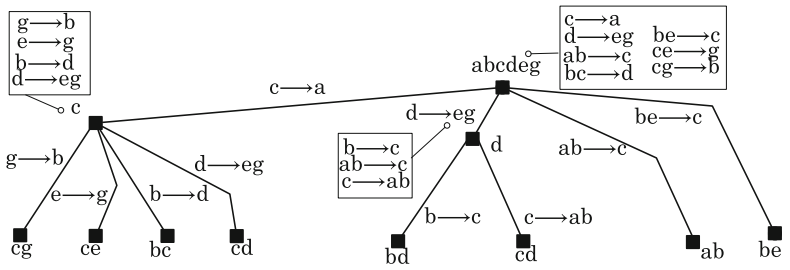

Fig. 3 $\mathbf{SL}_{\mathrm{FD}}$ Closure Keys Method example. Notice how the treatment of the implication set in each node produces a big reduction of the tree of Fig. 2 from 21 to 11 nodes

Moreover, the number of recursive calls is limited by the cardinality of Γ. The kernel of the method is the $\mathbf{SL}_{\mathrm{FD}}$ Closure presented in [12] providing a compound output, i.e., $\mathbf{SL}_{\mathrm{FD}}\mathit{Closure}(X, \Gamma) = \langle X^+, \Gamma' \rangle$.

This method produces all minimal keys because, in each step, it renders a closure X^+. If it is not the entire Ω, we go ahead by selecting a new antecedent from the implication set Γ'. We remark that this implication set has been depurated by means of $\mathbf{SL}_{\mathrm{FD}}$ Closure. By virtue of this, once we reached a leaf, the union of the antecedents in the way through conform a key and, therefore, the method is sound.

Concerning its completeness, any key in the system is produced since the search space induced by our method constitutes an exhaustive search. As a conclusion from the key study presented in [13], keys have to be composed with bricks being the antecedent of the implications. Those pruned branches do not fall down the completeness since the removed keys were redundant.

Algorithm 1: Closure_Keys

Data: Ω a set of attributes, Γ a set of implications, $\mathcal{C}$ an accumulator variable of a subset set of attributes, $\mathcal{K}$ a acumulator variable with sets of attributes.
begin
  $\mathcal{A}$:=Ω
  if $\Gamma = \varnothing$ **then** Add($\mathcal{K}$, $\{\mathcal{A} \cup \mathcal{C}\}$)
  else
  foreach $X \rightarrow Y \in [IM](\Gamma)$ **do**
    $< X', \Gamma' >$:=$\mathbf{SL}_{\mathrm{FD}}$Closure($X$, Γ)
    $\mathcal{C} := \mathcal{C} \cup X$
    Closure_Keys(X', Γ', $\mathcal{C}$, $\mathcal{K}$)
  /* Where [IM](Γ) denotes the implications in Γ being minimal FD. */
end

4 Implementation

The approach presented determines that, due to the construction of the search tree, parallel strategies and big hardware resources are strongly needed when we try to resolve problems containing substantial size at the input. In order to test the improvement, we are going to proceed with a parallel implementation of SST and Closure_Keys algorithms that works in two steps, (1) a partial method splits the entry problem into several subproblems and (2) a parallel method resolves all these generated subproblems and composes the final result, the minimal key set (see Fig. 4).

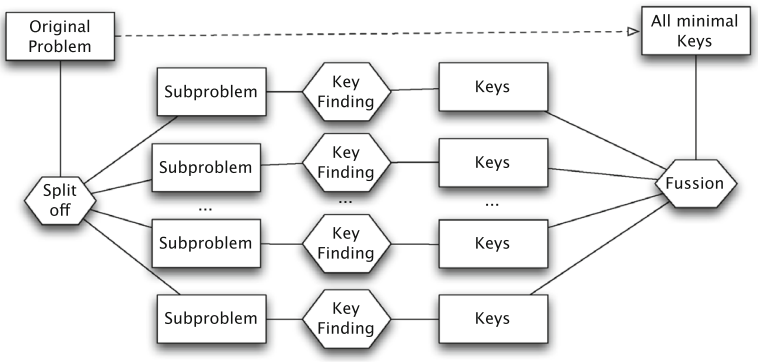

Fig. 4 Parallelism diagram

We can go along this kind of strategy over our parallel implementations because, due to the tableaux nature, each branch will be totally independent to others, so every node can be treated on its own. That is the great point of using parallelism within these techniques; we can deliver a huge number of problems to different cores so they can be solved simultaneously. By virtue of this, the size of the problems at the input could be greater without exceeding the machine limitations; at least, we have distanced the limit in a substantial way for the moment.

The application of the partial code moves us to split the entry problem into several subproblems. At this regard, we will stop at a certain level of the tree, and there, every node constitutes an independent problem by itself. Each of these problems represents a branch of the search tree and will be the input for the application of the parallel method, that will end up in a leaf of the tree. However, it is needed to go forward determining the value that will decide the splitting point, and this is, indeed, not an easy issue so far, as we discuss in the following paragraph.

As break-off value (BOV), determining the level in which the branch is considered an atomic to be treated by the parallel stage, we use the cardinal of the set of implications of the current node, since we have observed that, the bigger this cardinal is, the longer the branch *used to* reach. Once we have all these subproblems, and due to all of them conform an individual state of the algorithm, they are meant to be solved by the parallel code. However, if we are treating with big problems (those containing a substantial number of attributes and implications), the number of generated subproblems could be huge (we have run experiments with 50,000+ subproblems). Thus, handling them is a task only available to a significant amount of resources, as Supercomputing and Bioinnovation Center of the University of Málaga[1] supports us.

5 Experiments and results

We have randomly generated a set of experiments, varying the number of attributes and implications. We start with problems with 100 implications, each one built using 100 different attributes as well. Then, we move forward considering bigger problems

[1] http://www.scbi.uma.es.

Table 1 Attempts trying to improve execution times by increasing the number of cores

Problem and method	Implications 150 Partial$_t$ (s)	Attrib 150 Total$_t$ (s)	BOV 140 Nodes	Cores	Ratio
150150-4-*SST*	581	885	55.211	32	62
150150-4-*CK*	48	65	25.477	32	391
150150-4-*SST*	576	880	55.211	64	62
150150-4-*CK*	46	64	25.477	64	398

counting on 150 implications and 150 attributes. Notice that these numbers go far beyond machine capabilities, as it has been demonstrated in previous studies of this work [3]; they even substantially exceed the results given in [1], where parallelism techniques had already been applied.

Obviously, both methods obtain the same keys. Therefore, we have omitted this parameter in result tables since it only experimentally validates the method. Consequently, in order to compare SST method versus Closure Keys method, we focus on execution times and the number of nodes of the tree (so, we can figure out the size of the problem).

Number of nodes will always be the same for an individual experiment no matter the number of runs we do. However, execution times will not go along exactly with this statement. They could be slightly different due to it intrinsic nature. Differences concerning number of nodes will show how new methods have improved the algorithm as the depth of the tree has been sharply reduced, and consequently, execution times become better. Besides, we have included a last column in order to show the ratio of time and nodes between SST and CK. Regarding all these points, needless to say that every execution time shown here is the fruit of a post-statistical study [7,23] behind the results obtained from every run, so we kept the most reliable ones.

Before we reach the results section, the hardware architecture that has been used for developing every test shown in the paper can be visited in http://www.scbi.uma.es/site/scbi/hardware.

Every experiment we show here have been carried out using a cluster of 32 cores from those available within the hardware architecture mentioned above. Few subsequent experiments increasing the number of cores available went not up to expectations results in terms of execution times, as we briefly show in Table 1. Increasing the number of cores to engage parallelism within these kind of problems is a matter where much still remains to be investigated.

The first experiment starts solving a battery of hard load problems counting on 100 attributes and 100 implications. Results are shown in Table 2.

Even with such a big number of attributes and dependencies, it's been just around 10 min long for the slowest algorithm to finish (problem 100100-3). Yet, less than 300k nodes have been the maximum size of the Tableaux of all of them (problem 100100-7). It is absolutely out of mind thinking about solving these problems using sequential versions of the algorithms, since execution times would go out of hands.

Table 2 Parallel methods applied to big-size problems (I)

Problem and method	Attrib 100 Subp	Implications 100 Partial$_t$ (s)	BOV 90 Total$_t$ (s)	Cores 32 Nodes	Ratio
100100-1-*SST*	14	0	1	33	33
100100-1-*CK*	0	0	0	15	15
100100-2-*SST*	1.354	36	105	25.621	244
100100-2-*CK*	212	4	15	12.715	847
100100-3-*SST*	8.602	183	644	192.574	299
100100-3-*CK*	1.286	37	99	94.255	952
100100-4-*SST*	400	7	26	1.704	65
100100-4-*CK*	15	1	2	751	375
100100-5-*SST*	39	0	2	119	59
100100-5-*CK*	0	0	1	42	42
100100-6-*SST*	1.808	37	123	7.856	63
100100-6-*CK*	115	4	9	3.698	410
100100-7-*SST*	6.167	182	489	275.429	563
100100-7-*CK*	1.378	24	90	118.884	1.320
100100-8-*SST*	5.104	146	415	182.167	438
100100-8-*CK*	1.014	19	68	81.632	1.200
100100-9-*SST*	314	11	25	868	34
100100-9-*CK*	0	1	1	341	341
100100-10-*SST*	1.130	27	84	12.541	149
100100-10-*CK*	136	4	10	6.128	612

In fact, these are encouraging results since, earlier experiments developed using older methods [1] couldn't even have imagined to deal with such a huge problems.

This first experiment confirms which it was mentioned above: subproblems and nodes show a better performance by Closure Keys method.

A very remarkable outcome can be appreciated in problems 100100-{1,5,9}. Notice that Closure Keys does not create any subproblems. As this regard, the improvement is twofold: (1) in some cases, with the same BOV, the new method doesn't even need to move on the parallel implementation, partial one is enough to resolve these problems, and (2) to exploit this discovery, at the moment we need to deal with even more complex problems, we can establish a BOV nearer to the root of the tree and then, partial time will be significantly reduced.

Second experiment goes ahead giving another turn of the screw increasing the input of our experiments. We develop a new battery of 10 problems, increasing the number of implications and available attributes up to 150.

As far as these problems become more complex, the better is the improvement achieved by the new method. Executions times and number of nodes have been drastically reduced as shown in Table 3. Actually, we decided to add one more experiment (problem 150150-EXTRA) due to the specific remarkable results it reached.

Table 3 Parallel methods applied to big-size problems (II)

Problem and method	Attrib 150 Subp	Implications 150 Partial$_t$ (s)	BOV 140 Total$_t$ (s)	Cores 32 Nodes	Ratio
150150-1-*SST*	165	6	14	911	65
150150-1-*CK*	11	2	3	374	124
150150-2-*SST*	2.949	229	394	116.517	295
150150-2-*CK*	347	25	44	54.375	1.235
150150-3-*SST*	12.968	1.049	1.716	157.947	92
150150-3-*CK*	822	125	165	68.531	415
150150-4-*SST*	5.352	581	885	55.211	62
150150-4-*CK*	344	48	65	25.477	391
150150-5-*SST*	5.361	211	484	32.377	66
150150-5-*CK*	168	27	36	12.522	347
150150-6-*SST*	771	72	155	17.298	111
150150-6-*CK*	79	7	11	8.110	737
150150-7-*SST*	9.473	638	1.252	576.912	460
150150-7-*CK*	1.754	97	187	262.621	1.404
150150-8-*SST*	5.466	424	857	510.627	595
150150-8-*CK*	966	57	104	257.267	2.473
150150-9-*SST*	235	25	45	3.632	80
150150-9-*CK*	24	3	4	1.726	431
150150-10-*SST*	3.403	348	555	102.537	184
150150-10-*CK*	277	31	46	45.962	999
150150-EXTRA-*SST*	31.401	2.950	30.983	21,404,732	690
150150-EXTRA-*CK*	8.049	354	1.320	10,614,386	8.041

Several experiments are worth to be discussed separately. As a matter of fact, if we set our sight on problems 150150-{3,7}, number of subproblems generated are much less for Closure Keys method than SST. Difference is not trivial so far, it also guides us directly to a great reduction of partial execution times. Total execution times go along in the same direction. Regarding the size of the tree, the benefits by introducing the closure are pretty striking. Most of cases, the number of nodes becomes near to 50 % lower. Figure 5 would help us appreciating the differences more clearly.

6 Conclusions

We presented here a method to solve the key finding problem. Among the different strategies introduced in the literature, we follow the logic-based line pioneered in [20]. The new method presented here uses the Simplification Logic and introduces an alternative to the SST method presented in [5]. Here, we provide a further reduction of the tree size by shortening its depth, empirically confirmed by means of the number of

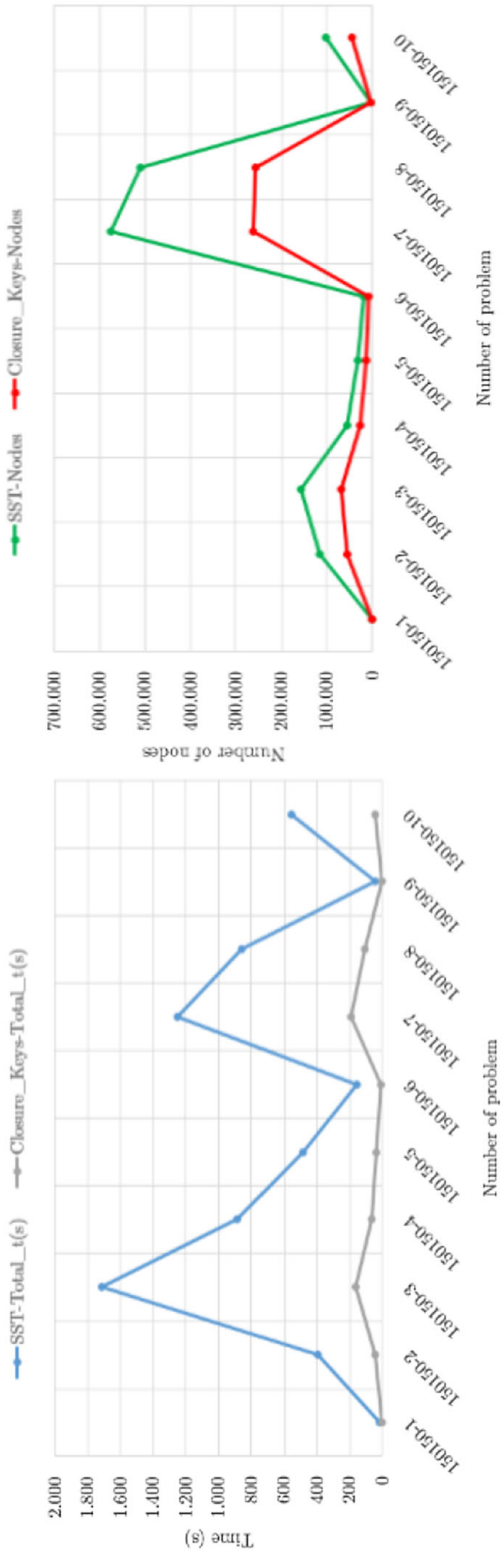

Fig. 5 Execution times and number of nodes for Big-size problems (II)

generated nodes. In addition, a parallel implementation has been developed following a map-reduce architecture. The inherent parallel design of the new method allows us to face problems containing a significant size at the input.

We have set two priorities for future works:

– Applying these mechanisms over real data stores, so we can figure out how they really could take advantage of parallelism working on these kind of datasets.
– The study of the impact concerning the number of cores involved in the experiments.

Acknowledgements Supported by Grants TIN2011-28084 and TIN2014-59471-P of the Science and Innovation Ministry of Spain, co-funded by the European Regional Development Fund (ERDF). The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the SCBI (Supercomputing and Bioinnovation) center of the University of Málaga (Andalucía Tech).

References

1. Benito-Picazo F, Cordero P, Enciso M, Mora A (2014) Increasing the efficiency of minimal key enumeration methods by means of parallelism. In: ICSOFT-EA 2014—proceedings of the 9th international conference on software engineering and applications, Vienna, Austria, 29–31 August 2014, pp 512–517
2. Codd EF (1970) A relational model of data for large shared data banks. Commun ACM 13(6):377–387
3. Cordero P, Enciso M, Mora A (2013) Automated reasoning to infer all minimal keys. In: Proceedings of the twenty-third international joint conference on artificial intelligence, IJCAI'13. AAAI Press, pp 817–823
4. Cordero P, Enciso M, Mora A, de Guzmán IP (2002) $\mathrm{Sl}_{\mathrm{fd}}$ logic: elimination of data redundancy in knowledge representation. In: Advances in artificial intelligence 8th Ibero-American conference on AI, Seville, Spain, November 12–15, pp 141–150
5. Cordero P, Enciso M, Mora A, de Guzmán IP (2014) A tableaux-like method to infer all minimal keys. Log J IGPL 22(6):1019–1044
6. Ganter B, Wille R (1997) Formal concept analysis: mathematical foundations, 1st edn. Springer-Verlag New York Inc, Secaucus
7. Goh TN (2010) An information management paradigm for statistical experiments. Qual Reliab Eng Int 26(5):487–494
8. Krajca P, Outrata J, Vychodil V (2008) Parallel recursive algorithm for FCA. In: 6th International conference on concept lattices and their applications (CLA), vol 433. CEUR WS, Olomouc, pp 71–82 (2008)
9. Lèvy G, Baklouti F (2005) A distributed version version of the ganter algorithm for general galois lattices. In: 3rd International conference on concept lattices and their applications (CLA), vol 162. CEUR WS, Olomouc, pp 207–221
10. Lucchesi CL, Osborn SL (1978) Candidate keys for relations. J Comput Syst Sci 17(2):270–279
11. Maier D (1983) The theory of relational databases. Computer Science Press, Rockville
12. Mora A, Cordero P, Enciso M, Fortes I, Aguilera G (2012) Closure via functional dependence simplification. Int J Comput Math 89(4):510–526
13. Mora A, de Guzmán IP, Enciso M, Cordero P (2011) Ideal non-deterministic operators as a formal framework to reduce the key finding problem. Int J Comput Math 88(9):1860–1868
14. Pernelle N, Sas F, Symeonidou D (2013) An automatic key discovery approach for data linking. Web Semant Sci Serv Agents WWW 23:16–30
15. Saiedian H, Spencer T (1996) An efficient algorithm to compute the candidate keys of a relational database schema. Comput J 39(2):124–132
16. Sismanis Y, Brown P, Haas PJ, Reinwald B (2006) Gordian: efficient and scalable discovery of composite keys. In: Proceedings of the international conference on very large data bases, pp 691–702
17. Sridhar R, Iyengar SS (1990) Efficient parallel algorithms for functional dependency manipulations. In: Proceedings of the 2nd international symposium on databases in parallel and distributed systems, DPDS '90, pp 126–137

18. Valtchev P, Duquenne V (2003) Towards scalable divide-and-conquer methods for computing concepts and implications. In: 4th International conference Journèes de l'Informatique Messine (JIM03): knowledge discovery and discrete mathematics, Metz (FR), pp 3–14
19. Valtchev P, Duquenne V (2008) On the merge of factor canonical bases. In: International conference on formal concept analysis (ICFCA). Lecture notes in computer science, vol 4933. Springer, Berlin, pp 182–198
20. Wastl R (1998) On the number of keys of a relational database schema. J Univers Comput Sci 4:547–559
21. Worland PB (2004) An efficient algorithm for 3nf determination. Inf Sci Inf Comput Sci 167(1–4):177–192
22. Yu CT, Johnson DT (1976) On the complexity of finding the set of candidate keys for a given set of functional dependencies. Inf Process Lett 5(4):100–101
23. Zobel J (1998) How reliable are the results of large-scale information retrieval experiments? In: Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '98, pp 307–314

Capítulo 4

Generadores Minimales

Delimitación..., ésta es una palabra a la que no temo, pues la labor de algo superior que posee el hombre, reside en un constante tender a limitar lo infinito, y en dividirlo y desintegrarlo en porciones perceptibles, es decir, en diferenciales.

Nosotros
Y. Zamiatin

Título:	Minimal generators, an affordable approach by means of massive computation
Autores:	Fernando Benito-Picazo, Pablo Cordero, Manuel Enciso, Ángel Mora
Revista:	The Journal of Supercomputing, Springer
Factor Impacto JCR:	1,532. Posición 44 de 103 (Q2)
Año:	2017
Categoría:	Computer Science, Theory & Methods
Publicación:	4 junio 2018
DOI:	10.1007/s11227-018-2453-z

J Supercomput
https://doi.org/10.1007/s11227-018-2453-z

Minimal generators, an affordable approach by means of massive computation

F. Benito-Picazo[1] · P. Cordero[1] · M. Enciso[1] · A. Mora[1]

Abstract Closed sets and minimal generators are fundamental elements to build a complete knowledge representation in formal concept analysis. The enumeration of all the closed sets and their minimal generators from a set of rules or implications constitutes a complex problem, drawing an exponential cost. Even for small datasets, such representation can demand an exhaustive management of the information stored as attribute implications. In this work, we tackle this problem by merging two strategies. On the one hand, we design a pruning, strongly based on logic properties, to drastically reduce the search space of the method. On the other hand, we consider a parallelization of the problem leading to a massive computation by means of a map-reduce like paradigm. In this study we have characterized the type of search space reductions suitable for parallelization. Also, we have analyzed different situations to provide an orientation of the resources (number of cores) needed for both the parallel architecture and the size of the problem in the splitting stage to take advantage in the map stage.

This work is partially supported by Project TIN2017-89023-P of the Science and Innovation Ministry of Spain, co-funded by the EU Regional Development (ERDF).

✉ A. Mora
amora@ctima.uma.es

F. Benito-Picazo
fbenito@lcc.uma.es

P. Cordero
pcordero@uma.es

M. Enciso
enciso@lcc.uma.es

[1] Universidad de Málaga, Málaga, Spain

Published online: 04 June 2018

Keywords Minimal generators · Formal concept analysis · Parallel methods · Logic

1 Introduction

Knowledge representation and reasoning are the two main pillars of artificial intelligence (AI). The growing interest in AI arisen in several areas beyond computer science is firmly rooted in the recent techniques to extract knowledge from the data and to be efficiently managed. One outstanding framework, based on a strong theoretical basis and also providing executable methods, is the Formal Concept Analysis (FCA) introduced by Ganter and Wille [11]. In FCA, data are stored in a table, representing a binary relation between G (a set of objects) and M (a set of attributes). FCA is not an approximate approach since it pursuits to capture, manage and analyze the complete knowledge from the information. This feature has a direct impact on the cost of the methods developed to extract and manage the information.

In this framework, two alternative knowledge representations are used: concept lattice and implications. The second one can be viewed as if-then rules already introduced in other areas, dressed with different clothes. Thus, in relational databases [4] they are named Functional Dependencies and in FCA they are named Implications [11]. All these notions capture the same idea (a pretty intuitive one): when some premise occurs, then a conclusion holds.

Since the size of the concept lattice (in the worst case) is $2^{\min(|G|,|M|)}$, the computational cost of the methods to build it has been considered as a limitation for the application of FCA. Algorithms to infer the concept lattice from the dataset were deeply studied and compared in [15]. Moreover, the extraction of the complete set of implications (basis) from a data set also presents an exponential behavior [5]. In both problems, the density of the dataset plays a major role in the performance of the execution of the method. To improve the computational behavior, in [14] the authors introduce the notion of redundant attributes to avoid the inclusion of such attributes in the extraction of the implications. Recently, in [8] a wide range of parallel methods to solve this second problem has been presented. These works motivate the need to combine some kind of reduction in the search space and a parallel execution to solve these complex problems.

In the two problems already mentioned, the input is the dataset and the output is its knowledge in terms of implications (basis) or attribute closed sets (concept lattice). However, here we deal with a complementary problem that allows to connect both knowledge representations: the enumeration of all the closed sets from a given set of implications. Moreover, we propose a method to produce not only all closed sets but also, for each of them, their canonical representations, named minimal generators.

Minimal generators are interesting not only from a theoretical point of view. In recent works, their computation is the core for solving other problems. The significance of minimal generators is well emphasized in the survey of Poelmans et.al. [21]: “Minimal generators have some nice properties: they have relatively small size and they form an order ideal”. Qu et al. [22] pointed out that decision implications involve to know all minimal generators.

Furthermore, they have been used as a key point to build basis, which constitutes a compact representation of the knowledge allowing a better performance of the reasoning methods based on rules. Missaoui et al. [17,18] present the use of minimal generators to compute basis involving positive and negative attributes whose premises are minimal generators. All the mentioned authors have considered the dataset as the input of the problem, namely, the minimal generators and closed sets are inferred from the plain data. In [10,20] some methods for solving this problem have been proposed.

In this work, we consider implications as elements to describe the information and we design a method to enumerate all closed sets and their minimal generators from this information, and not from the dataset. This is a complementary problem to the extraction of minimal generators from the dataset and, as far as we know, no previous work has been developed. In summary, we propose a method to transform the knowledge in terms of implications, into a more organized form so that a complete representation of the closed sets of attributes (and their minimal generators) is obtained. This complete and precise specification allows a further fast management of the semantics of the information contained in the dataset.

Our starting point is [6], where a logic-based method based on SL_{FD}, a sound and complete logic for implications was introduced. In that work, we present the MinGen algorithm which works by traversing the set of implications and applying a set of inference rules, building a search tree space. This shape of the search space limits its execution for medium-sized problems, because of the overwhelming resources of the sequential MinGen algorithm. Although the search space is suitable for a parallel extension of the method, some further research has to be carried out. On the one hand, we first consider the integration of some strategies to reduce the search space. In this work, we present two approaches, MinGenRd and GenMinGen, providing a single-level or a multi-level strategy, respectively. We also show how the MinGenRd is suitable for parallelization whereas GenMinGen is not. On the other hand, we approach a parallelization of the MinGenRd, rendering the MinGenRdPar algorithm. This new method is driven by a like-Map-Reduce strategy.

In addition to the MinGenRdPar method, we also provide a set of experiments in different situations to characterize the resources needed for the parallel execution and to establish a threshold in the splitting stage. In both issues, our goal is to look for an efficient execution of the method. We remark that we have used a random generation of synthetic problems and also real-world problems. Our intention is to show that our approach is valid in both situations.

The paper is organized as follows: after presenting the needed background in Sect. 2, we revisit in Sect. 3 our starting point: a previous logic-based method to enumerate all minimal generators and closed sets (`MinGen`). Then the reduction strategy by eliminating redundant branches is described in Sect. 4, introducing the MinGenRd and GenMinGen methods. Later, in Sect. 5, we design a parallel version of the MinGenRd algorithm, named MinGenRdPar, running on a massive computation architecture. We provide some stats of its performance together with an illustrative execution of the parallel algorithm over a real (and well-known) problem. We will end up with Conclusions and Future Works. Figure 1 illustrates the content of the paper.

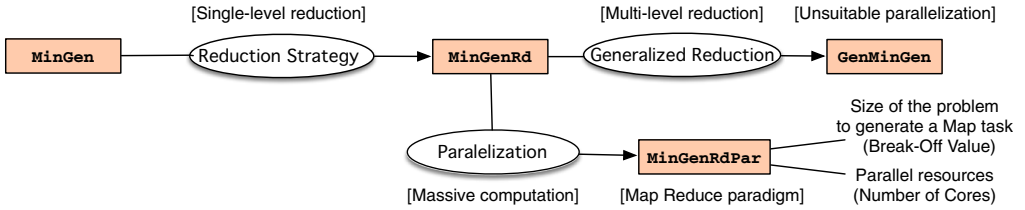

Fig. 1 Illustration of the content of the work

2 Preliminaries

Simplification Logic [7] is introduced by describing its four pillars: its *language*, its *semantics*, its *axiomatic system* and its *automated reasoning method*. Let M be a finite set of symbols (called *attributes*). The language over M is defined as

$$\mathcal{L}_M := \{A \to B \mid A, B \subseteq M\}$$

Formulas $A \to B$ are called (attribute) *implications* and the sets A and B are called *premise* and *conclusion* of the implication, respectively. Sets of implications are called *implicational systems*. In order to simplify the notation we omit the brackets in premises and conclusions and write their elements by juxtaposition. Thus, for instance, $ab \to cde$ denotes $\{a, b\} \to \{c, d, e\}$.

For introducing the semantics we use the notion of *closure operator*. For a more detailed description of closure operator see [23]. A closure operator on M is a mapping $c\colon 2^M \to 2^M$ that is *extensive*, *isotone* and *idempotent*. The fix-points for c are called *closed sets*. Closure operators are strongly connected to knowledge representation in different areas [1,3,9,12,16].

Since we use the logic as an executable tool, we have to substitute the semantic interpretation and inference for an efficient symbolic management. Thus, we introduce an axiomatic system considering reflexivity as axiom scheme

$$\texttt{[Ref]} \quad \frac{}{A \cup B \to A}$$

together with the following inference rules, called *fragmentation*, *composition* and *simplification*, respectively:

$$\texttt{[Frag]} \ \frac{A \to B \cup C}{A \to B} \qquad \texttt{[Comp]} \ \frac{A \to B,\ C \to D}{A \cup C \to B \cup D} \qquad \texttt{[Simp]} \ \frac{A \to B,\ C \to D}{A \cup (C \setminus B) \to D}$$

This axiomatic system is *sound* and *complete* (both semantic and syntactic derivations coincide). This result allows us to design automated reasoning methods. These methods can be approached by using a new closure operator called *syntactic closure*: give an implicational system $\Sigma \subseteq \mathcal{L}_M$, a set $X \subseteq M$ is said to be closed w.r.t Σ if $A \subseteq X$ implies $B \subseteq X$ for all $A \to B \in \Sigma$. Since M is closed w.r.t Σ and any intersection of closed sets is closed, we can define the following closure operator:

$$(\,)^+_\Sigma\colon 2^M \to 2^M \qquad X^+_\Sigma = \bigcap \{Y \subseteq M \mid Y \text{ is closed w.r.t } \Sigma \text{ and } X \subseteq Y\}$$

The automated reasoning method in Simplification Logic is based on the Deduction Theorem and three syntactic equivalences which allow us to transform the set of

implications with no semantic loss, i.e., the meaning of the knowledge representation is preserved.

Theorem 1 (Deduction Theorem) *Let* $A \rightarrow B \in \mathcal{L}_M$ *and* $\Sigma \subseteq \mathcal{L}_M$. *Then,*

$$\Sigma \vdash A \rightarrow B \quad \textit{iff} \quad B \subseteq A^+_\Sigma \quad \textit{iff} \quad \{\varnothing \rightarrow A\} \cup \Sigma \vdash \{\varnothing \rightarrow B\}$$

The following corollary characterizes the closed sets built with the syntactic closure as a maximum set enclosing the information of implications.

Corollary 1 *Let* $\Sigma \subseteq \mathcal{L}_M$. *For all* $X \subseteq M$, *one has* $X^+_\Sigma = \max\{Y \subseteq M \mid \Sigma \vdash X \rightarrow Y\}$.

In [19] we present a novel algorithm to compute closures using Simplification Logic. Given a set $A \subseteq M$, to compute A^+_Σ, the formula $\varnothing \rightarrow A$ is added to Σ and used as a seed by the reasoning method by using the equivalences provided by the previous proposition. Specifically, the algorithm uses the following equivalences:

- **Eq. I**: If $B \subseteq A$ then $\{\emptyset \rightarrow A, B \rightarrow C\} \equiv \{\emptyset \rightarrow A \cup C\}$.
- **Eq. II**: If $C \subseteq A$ then $\{\emptyset \rightarrow A, B \rightarrow C\} \equiv \{\emptyset \rightarrow A\}$.
- **Eq. III**: Otherwise $\{\emptyset \rightarrow A, B \rightarrow C\} \equiv \{\emptyset \rightarrow A, B \setminus A \rightarrow C \setminus A\}$.

One outstanding characteristic of our algorithm, named as Function `Cls`, is that, besides the attribute set corresponding to the closure, it also renders a set of implications corresponding to the complementary knowledge that describes the information which is not within the closure. This new set could be further treated in an iterative process, as our Minimal Generator algorithm will do. An illustrative execution of this function is shown in Example 1.

Example 1 Let $\Sigma = \{a \rightarrow c, bc \rightarrow d, c \rightarrow ae, d \rightarrow e\}$ and $A = \{c, e\}$. The algorithm `Cls` returns $A^+ = \{a, c, e\}$ and the following table summarizes its trace:

$Guide$	Σ
$\emptyset \rightarrow ce$	$a \rightarrow c$ $bc \rightarrow d$ $c \rightarrow ae$ $d \rightarrow e$
$\emptyset \rightarrow ce$	$a \rightarrow \not{c}$ $b\not{c} \rightarrow d\not{c} \rightarrow a\not{c}$ $d \rightarrow \not{c}$
$\emptyset \rightarrow ace$	$b \rightarrow d$

Therefore, $\texttt{Cls}(\{c, e\}, \{a \rightarrow c, bc \rightarrow d, c \rightarrow ae, d \rightarrow e\}) = (\{a, c, e\}, \{b \rightarrow d\})$.

An implicational system is said to be *complete for a closure operator* if it captures all the knowledge relating to that operator, that is to say, the implicational system expresses in the language of logic all knowledge relative to it. Therefore, when an implicational system Σ is complete for a closure operator c, its syntactic closure coincides with c, i.e., $X^+_\Sigma = c(X)$ for all $X \subseteq M$.

To efficiently manage all this knowledge, we characterize it with all the closed sets and their minimal generators. That is, given a set of implications Σ, closed sets characterize attribute sets with maximal meaning and, looking for a better integration in applications, in addition to the closed sets it would be very appreciated to provide a

compact representation of these closed sets. In other words, enumerate for each closed set those subsets that generate them. The following definition comes to formalize these kind of subsets.

Definition 1 Let Σ be a set of implications, $()^+_\Sigma \colon 2^M \to 2^M$ be its closure operator and $C \subseteq M$ a closed set, i.e., $(C)^+_\Sigma = C$. The set $A \subseteq M$ is said to be a minimal generator (mingen) for C if $(A)^+_\Sigma = C$ and, for all $X \subseteq A$, if $(X)^+_\Sigma = C$ then $X = A$.

As we mentioned before, the generation of the implicational system associated with a closure operator is a hard problem and the reverse one has also an exponential behavior. Thus, here we deal with the definition of a method to solve this problem (Sect. 3) and also with the design of its efficient implementation (Sect. 5), particularly approaching the problem by using parallelism.

3 Minimal generators method

In [6], the Simplification Logic was used as the tool to find all the minimal generators (mingens) from a set of implications. The method applies the Function `Cls` to guide the search of new minimal generator candidates. Specifically, given a set of attributes M and an implicational system Σ, the algorithm renders a mapping $mg_\Sigma \colon 2^M \to 2^{2^M}$ that satisfies the following:

$\forall X, Y \subseteq M$, $X \in mg_\Sigma(C)$ iff C is closed for $(\,)^+_\Sigma$ and X is a mingen for C.

Example 2 For the implicational system introduced in Example 1, the mapping mg_Σ is described as follows:

X	$\varnothing$	b	e	be	de	ace	bde	$acde$	$abcde$
$mg_\Sigma(X)$	$\varnothing$	b	e	be	d	a	bd	ad	ab
						c		cd	bc

Otherwise, X is not closed and $mg_\Sigma(X) = \varnothing$. Notice that $\varnothing$ is closed and $mg_\Sigma(\varnothing) = \{\varnothing\}$, i.e., $\varnothing$ is a minimal generator of the closed set $\varnothing$.

The algorithms we introduce in this section need to use the following operation for this kind of mappings. Given two mappings $mg_1, mg_2 \colon 2^M \to 2^{2^M}$, the mapping $mg_1 \sqcup mg_2 \colon 2^M \to 2^{2^M}$ is defined as

$$(mg_1 \sqcup mg_2)(X) = \texttt{minimals}(mg_1(X) \cup mg_2(X)) \quad \forall X \subseteq M$$

Thus, $Y \in (mg_1 \sqcup mg_2)(X)$ if and only if Y is a minimal set of $mg_1(X) \cup mg_2(X)$ in $(2^M, \subseteq)$, i.e., $Y \in mg_1(X) \cup mg_2(X)$ and there does not exist another attribute set $Z \in mg_1(X) \cup mg_2(X)$ such that $Z \subsetneq Y$.

We already have all the tools needed to define the Minimal Generator method. It was introduced in [6] and here we describe it as Function MinGen depicted below, together with an illustrative example of its application (see Example 3).

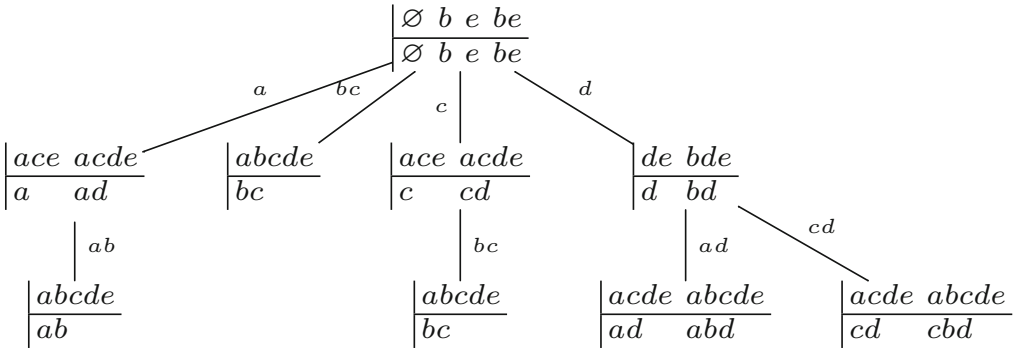

Fig. 2 Search tree for MinGen in Example 3

Function MinGen(M,Label,Guide,Σ)

input : M, the set of all attributes;
Label, an auxiliar set to build a minimal generator;
Guide, an auxiliar set to build a closed set;
Σ, an implicational system on M;
output: The mapping mg_Σ
begin
  foreach $X \subseteq M$ **do**
    $mg_\Sigma(X) := \varnothing$ (Guide,Σ):=Cls(Guide,Σ);
  $M := M \setminus$Guide;
  Premises := $\{A \subseteq M \mid A \to B \in \Sigma$ for some $B \subseteq M\}$;
  ClosedSets:= $\{X \subseteq M \mid A \not\subseteq X$ for all $A \in$ Premises$\}$;
  foreach $X \in$ ClosedSets **do**
    mg_Σ(Guide $\cup X$) := {Label $\cup X$}
  foreach $A \in$ Premises **do**
    $mg_\Sigma := mg_\Sigma \sqcup$ MinGen(M, Label $\cup A$, Guide $\cup A$, Σ)
  return mg_Σ

Example 3 For the implicational system introduced in Example 1, the Function MinGen ($abcde, \varnothing, \varnothing, \{a \to c, bc \to d, c \to ae, d \to e\}$) returns

X	$\varnothing$	b	e	be	ace	$acde$	$abcde$	de	bde
$mg_\Sigma(X)$	$\varnothing$	b	e	be	a	ad	ab	d	bd
					c	cd	bc		

The search tree is shown in Fig. 2.

4 Reducing the search space in the MinGen method

In this subsection, we present a further MinGen method corresponding to a significantly enriched version of the original one. We have integrated a reduction to avoid the generation of redundant minimal generators and closed sets. The aim of this pruning is to identify redundant branches in the search space to avoid their exploration. This reduction must only be carried out if we can ensure that all the information regarding minimal generators has been collected in other branches. In Function MinGenRd this reduction strategy is implemented in line #1. Thus, to ensure that the information generated in one branch is superfluous, we design a prune based on set inclusion involving

all the nodes at the same level. We illustrate how this technique works in the following example:

Example 4 Given the search tree previously introduced in Example 3, the Function MinGenRd $(abcde, \varnothing, \varnothing, \{a{\rightarrow}c, bc{\rightarrow}d, c{\rightarrow}ae, d{\rightarrow}e\})$ applies a reduction strategy avoiding to open the branch whose label is a superset of another edge at the same level. Particularly, the branch labeled bc in Fig. 2 will not be opened because it is not minimal in the set $\{a, bc, c, d\}$.

Function MinGenRd(M,Label,Guide,Σ)

output: The mapping mg_Σ
begin
  foreach $X \subseteq M$ **do**
    $mg_\Sigma(X) := \varnothing$ (Guide,Σ):=Cls(Guide,Σ);
  $M := M \setminus$Guide;
[#1] Premises := Minimals$\{A \subseteq M \mid A \rightarrow B \in \Sigma$ for some $B \subseteq M\}$;
  ClosedSets:= $\{X \subseteq M \mid A \nsubseteq X$ for all $A \in$ Premises$\}$;
  foreach $X \in$ ClosedSets **do**
    mg_Σ(Guide $\cup X$) := {Label $\cup X$}
  foreach $A \in$ Premises **do**
    $mg_\Sigma := mg_\Sigma \sqcup$ MinGenRd(M, Label $\cup A$, Guide $\cup A$, Σ)
  return mg_Σ

4.1 A generalization of the reduction strategy

In this section, we propose a generalization of the reduction strategy by considering the subset inclusion test not only with the nodes at the same level, but with all the minimal generators computed before the opening of each branch. One step further is to take advantage of the minimal generators already computed to increase the number of branches not needed to be opened. As Function GenMinGen shows, we consider this generalized pruned in #1, and then, the list of minimal premises to be considered in each stage is built in #2.

Function GenMinGen(M,Label,Guide,Σ,MinGenList)

output: The mapping mg_Σ
begin
  foreach $X \subseteq M$ **do**
    $mg_\Sigma(X) := \varnothing$ (Guide,Σ):=Cls(Guide,Σ);
  $M := M \setminus$Guide;
  Premises := Minimals$\{A \subseteq M \mid A \rightarrow B \in \Sigma$ for some $B \subseteq M\}$;
  ClosedSets:= $\{X \subseteq M \mid A \nsubseteq X$ for all $A \in$ Premises$\}$;
  foreach $X \in$ ClosedSets **do**
    mg_Σ(Guide $\cup X$) := {Label $\cup X$}
  foreach $A \in$ Premises **do**
[#1] **if** *there no exists* $Y \in$ MinGenList *such that* $Y \subseteq A$ **then**
      $mg_\Sigma := mg_\Sigma \sqcup$ GenMinGen(M, Label $\cup A$, Guide $\cup A$, Σ, MinGenList)
[#2] **add** A **to** MinGenList;
  return (mg_Σ)

Notice that in Fig. 2 the branches labeled with ad and cd will now not be opened because in the previous level labels a and c were (respectively) opened. In the output of Function GenMinGen the minimal generators ad and cd appears in previous branches whereas abd and cbd are not computed because they are not really minimal generators (see the enumeration closed sets and minimal generators for this problem in Example 3).

4.2 Testing the performance of the reducing techniques

Once we have presented the original Minimal Generator method (MinGen) along with the two approaches (MinGenRd and GenMinGen) and an illustration of their search spaces has been shown, we present now a global comparison of the performance achieved by each of them.

For that matter, we have developed the corresponding implementations to apply the methods to a battery of sets of implications randomly generated. To evaluate this comparison, two different metrics are applied: (1) the execution time of the algorithm and (2) the number of nodes in the search tree built by the method. The reason for this selection becomes reasonable as follows. The execution time arises as the classical measure to test the performance, but it is always hardly linked to the resources we are working with. So, we add the number of nodes of the tree as a metric to compare these algorithms since it could better guide us to put forward an argument to decide which algorithm is the best as it is an architecture-independent value.

In addition, due to the intrinsic nature of execution time, it is also imperative to state that every experiment shown throughout this document has been repeated several times, so we can now write down the most reliable average values w.r.t. the execution time metric.

In the experiments, the hardware configuration used is: Intel(R) Core(TM) i7-6700HQ CPU 2.60 GHz, 8 Gb RAM memory, running over Windows 10. We have generated a battery of different inputs to be used in the sequential implementation to show the improvements of the methods, that is:

- A synthetic test. Contains 5 testing files with 50 implications built using 50 possible different attributes. The implications are randomly generated.
- A real dataset. We focused on MovieLens datasets[1] which have been widely used in education, research, and industry [13]. In this spirit, we chose the MovieLens10M dataset which is a dataset totally accessible through MovieLens web page and contains huge information about movies, genres, ratings, users, etc. From all this information, we are going to generate a table with 10.681 rows, where we face every movie with all the possible genres included in the dataset. The extraction of implications from the dataset renders a set of 19 attributes (genres) and a set of 245 implications.

Having said that, the results of the experiments are shown in Fig. 3. Giving an overall view of the results, it can be clearly seen how the reduction strategies significantly reduce both the number of nodes and the execution times. Indeed, experiment

[1] https://grouplens.org/datasets/movielens/.

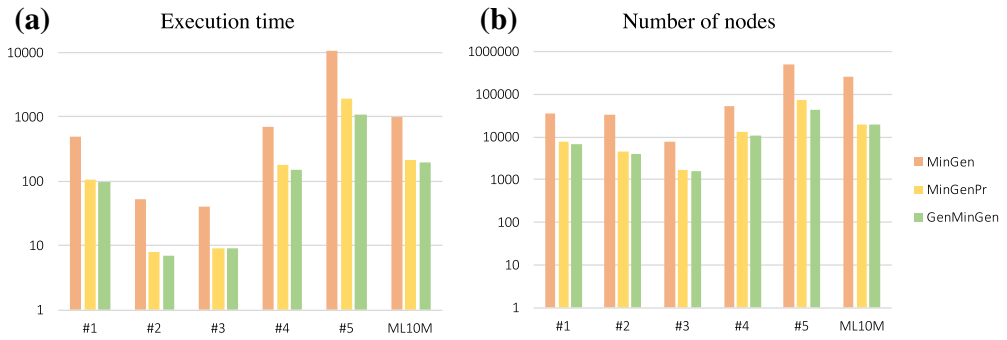

Fig. 3 Execution times (**a**) and number of nodes (**b**) results for the sequential experiments. Notice the logarithmic scale applied to the axes to better view the results

over 'sequential-5' file shows how both metrics have been drastically reduced. The execution time has been reduced from more than 10.000 s to less than 2.000 s. Besides, it is also worth to mention the reduction of the number of nodes which implies less resources needed in terms of memory and storage. Overall, like it was meant to happen, MinGen is surpassed by MinGenRd, and GenMinGen improves the latter as well.

Although GenMinGen has stated to have a better performance than MinGenRd (and both of them better than MinGen), the former is not suitable for parallelization. This is because it demands communication among all the subproblems previously obtained. This breaks our parallel philosophy, avoiding the use of massive computation. For this reason, we have studied a parallelization of the MinGenRd method.

5 Parallel computation of minimal generators

We have already shown the improvements reached by the reduction strategies within the minimal generators methods. However, once it comes the case that we want to use these methods over larger inputs, we can figure out in the light of the results obtained, that execution times of the sequential methods would go too far to be easily handled. Nonetheless, taking into account that every branch of the tree created by the methods constitutes a problem on its own, we can think about resolving them simultaneously and, later, combining the partial solutions to get the final output. This natural view of the problem as a parallel execution, provided by our logic-based methods, leads us to develop a new and parallel version of the minimal generators method.

Before going further, we introduce that the supercomputing resources and architecture that have been used to run the following parallel experiments are those provided by the Supercomputing and Bioinnovation Center of the University of Málaga.[2] In particular, we have developed each experiment using 32 nodes cluster SL230, counting on 16 cores and 64GB RAM memory and 7 nodes cluster DL980, counting on 80 cores and 2 TB RAM memory. Communications are carried out on Infiniband Net FDR and QDR. These cores are reserved just for our use so we can assess reliable results regarding execution times.

[2] https://www.scbi.uma.es/.

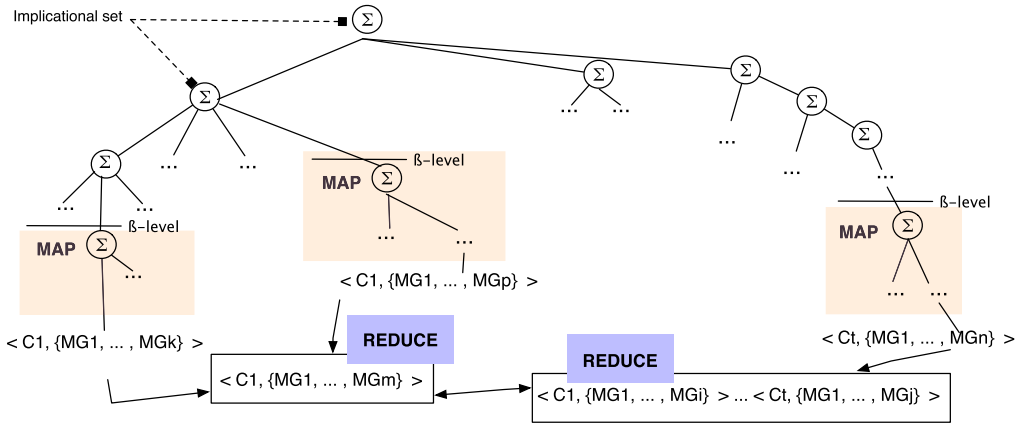

Fig. 4 Illustration of the map-reduce paradigm used in the design of MinGenRdPar

5.1 Parallel algorithm

We have developed a parallel implementation of the methods following the framework shown by the authors in [2]. In summary, it takes advantage of massive computation following the map-reduce paradigm. This parallel implementation of the methods performs in two stages. First one is the splitting stage. There, a single core builds the search space until it reaches a certain depth level of the tree. The goal is to split the original problem into several problems that can be treated by multiples cores, playing the role of a "Map" procedure. It goes this way until the size of the current node becomes lower than a given size (β) w.r.t the number of implications. From that moment on, each of these subproblems is resolved in parallel using multiple cores until we reach the leaves of the tree. Finally, a single core is used to compose the global result by merging the different outputs of each subproblems into a final one. This becomes necessary in order to eliminate redundancies to obtain the *minimal* generators. It is in this stage where the Reduce procedure acts as a filter to avoid redundant information in the global output. The architecture of our approach is depicted in Fig. 4.

In addition, we show the pseudocode of our parallel $MinGenRd$ algorithm in Function MinGenRdPar. We have emphasized the main stages of our implementation that likely correspond to the Map-Reduce paradigm.

The β level, named BOV (Break-Off Value) in the algorithm, is a critical parameter since it is in charge of deciding when to split the original problem into subproblems to manage them in parallel, generating a new Map task. As we will broadly explain later in Sect. 5.3, a hard experimental study has been needed to determinate the most suitable value for BOV.

Function MinGenRdPar(M,Label,Guide,Σ,BOV)

output: The mapping mg_Σ
begin
  foreach $X \subseteq M$ **do** $mg_\Sigma(X) := \varnothing$ (Guide,Σ):=Cls(Guide,Σ);
  $M := M \setminus$Guide;
  Premises := Minimals$\{A \subseteq M \mid A \to B \in \Sigma$ for some $B \subseteq M\}$;
  ClosedSets:= $\{X \subseteq M \mid A \nsubseteq X$ for all $A \in$ Premises$\}$;
  foreach $X \in$ ClosedSets **do** mg_Σ(Guide $\cup X$) := {Label $\cup X$}
  If $\mid \Sigma \mid \leq BOV$ **then do in parallel**
    // [MAP]
    foreach $A \in$ Premises **do**
      // [REDUCE]
      $mg_\Sigma := mg_\Sigma \sqcup$ MinGenRd(M, Label $\cup A$, Guide $\cup A$, Σ)
    return mg_Σ
  else
    // [Splitting stage]
    foreach $A_k \in$ Premises **do**
      $mg^k_\Sigma =$ MinGenRdPar(M, Label $\cup A_k$, Guide $\cup A_k$, Σ, BOV)
    forall the mg^k_Σ **do**
      // [REDUCE]
      $mg_\Sigma := mg_\Sigma \sqcup mg^k_\Sigma$;
    return mg_Σ

5.2 Sequential versus parallel experiments

This time, we have carried out an experiment to test the performance of the parallel implementation MinGenRdPar versus the sequential implementation MinGenRd. The input files we are going to use will count on numbers raised up to 150 attributes and 150 implications. Even with these numbers, three times higher than the sequential experiment in Sect. 4.2, parallel version obtains results in an admissible time as we can notice on the results given in Table 1. Looking closely, Table 1 collects the following information from left to right: (1) identifier of the input file and the method used to resolve, (2) execution time of the sequential version, (3) number of subproblems generated by the splitting stage of the parallel implementation, (4) execution time of the splitting stage, (5) execution time of parallel resolution, (6) execution time of the whole process, (7) number of nodes of the generated tree and (8) number of minimal generators obtained.

In conclusion, thanks to the application of the parallel implementation, we have been able to reduce the execution times from hours and days (experiments over #-{2,3,7,8,10}-sequential) down to just a few minutes. In other words, parallel computation improves the execution times even when dealing with big-sized problems.

5.3 Estimation of the BOV

There is a crucial aspect we have to keep in mind in the first stage of the parallel implementation, i.e., the splitting. We need to decide when a subproblem has to be generated (Map procedure) to be simultaneously executed. To this end, we are going to

Table 1 Comparison between both sequential and parallel versions of MinGenRd algorithm applied to big-sized problems

Problem	Seq.time (s)	Subp	Split_t (s)	Parallel_t (s)	Total_t (s)	Nodes	MinGens
#1	***43***	11	3	1	*4*	374	216
#2	***17.352***	347	220	55	*275*	54.375	6.273
#3	***33.338***	822	2.338	251	*2.589*	68.531	6.529
#4	***4.612***	344	350	97	*447*	25.477	2.478
#5	***1.585***	168	432	30	*462*	12.522	1.159
#6	***1.653***	79	35	7	*42*	8.110	1.436
#7	***107.238***	1.754	958	242	*1.200*	262.621	9.113
#8	***61.381***	966	253	188	*441*	257.267	5.538
#9	***372***	24	7	2	*9*	1.726	683
#10	***7.484***	277	186	65	*251*	45.962	2.969

use a value, denominated BOV, which represents the cardinal of the set of implications of the current node, since we have empirically observed that as long as this cardinal grows, the longer the branch of the tree uses to reach. The BOV is computed as a percentage of the size of the input that has to be estimated to balance the work done by each core.

Yet, deciding the BOV is a decisive task of the current investigation due to the following difficulties. On the one hand, if we decide to stop at a level near to the root of the tree by selecting a low BOV (i.e., a high number of implications), we are certainly reducing the execution time of the splitting stage and only a few subproblems would be created. Accordingly, since the tree would not have been able to spread out yet, then we will not have enough material to be managed in parallel using different cores. On the other hand, if we stop the split in a depth level far from the root, the splitting process will surely create a large amount of subproblems but its execution time would surely grow up. This is why we have selected a BOV empirically, as it is really difficult to get the right value by just analyzing the input theoretically. However, after making a lot of experiments, several aspects worth to be mentioned and this section stands to this effect.

We have hitherto established a BOV of 140 (recall that our number of implications is 150) within the experiments as it has shown to be the best one to leverage parallelism based on our experience. That is to say, we are choosing a BOV that corresponds to the $\approx$ 93.33% of the original set of implications. However, in order to explore this fact, we have repeated our parallel experiments using different BOVs. We take lower BOVs that makes the splitting stage to go deeper into the tree, specifically, we will use both 130 and 100 BOVs, i.e., the $\approx$ 86 and $\approx$ 66% of the whole original set of implications, respectively. This particular selection of values is not made by chance but with the aim of conveying several situations we have found after carrying out plenty of experiments which we come to analyze next. Results concerning execution times are gathered in Table 2.

Table 2 Experiments using different BOV within the parallel version of MinGenRd algorithm applied to big-sized problems

Problem	Subp	Split$_t$ (s)	Parallel$_t$ (s)	Total$_t$ (s)	BOV (%)
#1	0	44	–	**44**	**66.67**
	0	41	–	***41***	***86.67***
	11	3	1	*4*	*93.33*
#2	0	18.533	–	**18.533**	**66.67**
	885	8.532	58	***8.590***	***86.67***
	347	220	55	*275*	*93.33*
#3	0	33.377	–	**33.377**	**66.67**
	0	33.285	–	***33.285***	***86.67***
	822	2.338	251	*2.589*	*93.33*
#4	0	4.858	–	**4.858**	**66.67**
	308	3.703	22	***3.725***	***86.67***
	344	350	97	*447*	*93.33*
#5	0	1.601	–	**1.601**	**66.67**
	0	1.547	–	***1.547***	***86.67***
	168	432	30	*462*	*93.33*
#6	0	772	–	**772**	**66.67**
	144	492	11	***503***	***86.67***
	79	35	7	*42*	*93.33*
#7	0	167.451	–	**167.451**	**66.67**
	5.412	96.433	295	***96.728***	***86.67***
	1.754	958	242	*1.200*	*93.33*
#8	0	75.060	–	**75.060**	**66.67**
	5.344	41.404	375	***41.779***	***86.67***
	966	253	188	*441*	*93.33*
#9	0	82	–	**82**	**66.67**
	24	42	2	***44***	***86.67***
	24	7	2	*9*	*93.33*
#10	0	9.438	–	**9.438**	**66.67**
	697	6.569	50	***6.619***	***86.67***
	277	186	65	**251**	**93.33**

There are several outcomes within that table. First and foremost, except for problem #1 where there exist a subtle difference, all the problems behave worse regarding the execution time when we delay the splitting point. To find out an explanation, we put our attention in the number of generated subproblems where four different behaviors arise when we vary the BOV. We proceed to enumerate each of them with the support of the results obtained in Table 2.

Let $BOV_1 > BOV_2$, and let Sp_1, Sp_2 the number of generated subproblems associated with each of them. So, there might be situations where:

- $Sp_1 < Sp_2$. The algorithm has more scope to expand the tree and generate more subproblems. In this way, parallelism would take advantage but the time needed to split scupper the global execution time. Problems #{2,6,7,8,10} reflect this situation.
- $Sp_1 > Sp_2$. It can happen that even going deeper into the tree we obtained less nodes, i.e., less subproblems. This is because there are branches that end before reaching the BOV and they are resolved within the split stage, so the execution time grows higher. We can see this case in problem #4.
- $Sp_1 \neq 0 \wedge Sp_2 = 0$. Following on from the previous point, we can fall in an extreme situation where no subproblems are generated within the split stage because every branch of the tree ends up before the splitting point. This situation may be considered as the worst case since the parallel implementation performs the same as the sequential one. This happens for every problem when using a $BOV \approx 66\%$ and also in problems #{1,3,5} even with a $BOV \approx 86\%$.
- $Sp_1 = Sp_2$. This reflects an entangled situation in which lower levels of the tree may count on the same number of nodes than the higher ones. The point is that when we go deeper in the tree we may be generating more nodes as the tree expands and so the number of subproblems would grow up, but also, it may be several branches that end before going deeper and so it reduces the number of nodes. Therefore, with this additions and subtractions, the global calculation of subproblems can end up in a draw with different BOVs as shown in problem #9. But that's not all. Problem #9 indeed shows the same number of subproblems for $BOV \approx 93.33\%$ and $BOV \approx 86\%$, however, execution time is not the same, actually, it is worse for a $BOV \approx 86\%$ since we have stretched on the split stage.

As we have already mentioned before and after the analysis shown in this part, it becomes clear that trying to infer the best BOV for an experiment is not an easy issue so far as it depends on many possible situations we can face when dealing with the minimal generators enumeration problem.

5.4 Estimation of the suitable number of cores

Up to now, we can think it is just a matter of resources that we can increase the input size and still get results within a reasonable time. However, it is not so obvious and this subsection is dedicated to analyze this fact. For this purpose, the big-sized problems used before in Sect. 5.2 will be tested again by using 16, 32, 48, 64 and 80 cores each time. Results are shown in Table 3.

This is not up to expectations results. Although resources are better now, results do not come along. It can be seen that, we do improve the performance by using more cores (e.g., from 16 to 64 cores), however, there comes a moment where no profit is obtained; results are pretty similar for 48, 64 and 80 cores. This is due to the fact that there are several branches in the tree (maybe just one) that take so long to finish and then, no matter how much we improve the resources, the global time remains almost the same. The problem is that we cannot predict for the time being whether a branch will result in a long or a short one. This situation stalls our first intentions of blindly growing the amount of resources and shows up that, the number of cores for

Table 3 Execution times (in seconds) results obtained when increasing the number of cores applied

Problem	Number of cores				
	16	32	48	64	80
#1	5	4	*3*	*3*	*3*
#2	310	275	199	*190*	197
#3	2.742	2.589	2.122	2.078	*2.075*
#4	512	447	444	*440*	446
#5	598	462	*416*	445	442
#6	50	42	*34*	35	*34*
#7	1.457	1.200	1.078	*1.010*	1.012
#8	499	441	432	430	*425*
#9	9	9	*7*	*7*	*7*
#10	267	251	196	*194*	195

Table 4 Parallel minimal generator algorithm applied to a real-world dataset

Problem and Method	Subp	Split_t (s)	Parallel_t (s)	Total_t (s)	Nodes	MinGens
Mushrooms-parallel	224	152	9	161	81.363	17.127

this problem can be established to 64 as an optimal value, achieving a balance between resources needed and benefits obtained.

5.5 Real-world dataset experiment

To properly finalize this section, we bring now the results obtained when applying our algorithm to a real-world dataset. In particular, we put our focus in the Mushroom Data Set[3] accessible from the website of the University of California, Irvine (UCI).[4] Basically, this dataset includes descriptions of hypothetical samples corresponding to 8.124 species of gilled mushrooms using 22 attributes. On this dataset, an adaptation is made in order to convert multi-valued information into binary information. As a result, we will manage a dataset with 126 attributes. The number of implication inferred from this dataset is 1.587. We remark that with this number of implications, sequential version of the algorithm does not finish.

To carry out this experiment we use the conclusions reached by the previous sections. Therefore, we will use 64 different cores and a $BOV \approx 93.33\%$, i.e., $BOV = 1.481$, as they seem to be the best values to proceed. The results of this experiment are shown in Table 4 where it clearly arises how our algorithm fulfills when dealing with a big-sized and real-world dataset.

[3] https://archive.ics.uci.edu/ml/datasets/mushroom.

[4] http://archive.ics.uci.edu/ml/.

6 Conclusions and future works

Enumerating all the closed sets and their minimal generators is a hard problem, but essential in several areas and an opportunity to show the benefits of FCA for real applications. Such information can be obtained from a dataset or from a set of implications. In this work we focus on this second problem, which have been approached in a lesser extent. In order to face this task, this work presents an efficient reduction of the search space technique to improve the performance of minimal generator enumeration. The new method has been designed to fit the Map-Reduce architecture. Here is where parallel computation comes to make it possible for us to deal with such amount of information.

In particular, we have designed two new methods (MinGenRd and GenMinGen) implementing pruning strategies to reduce the search space. The empirical study proves how these methods surpass the previous one in the literature (MinGen). We also establish the adequacy of MinGenRd on being implemented following a parallel implementation by means of the Map-Reduce paradigm. Therefore, as the main contribution of this work, we propose the parallel implementation of MinGenRd method. The empirical study proves the very significative improvement achieved w.r.t the original sequential version. The parallel methods to compute minimal generators can make really usable these methods in practical applications.

To properly characterize the parallel approach, we have developed two battery of experiments: (1) we have established the BOV threshold, a parameter included in the algorithm to effectively apply the Map procedure taking the most of this paradigm and, (2) we have established the number of cores need to execute this method, thus delimiting the resources of the parallel architecture.

As future works, we plan to study how to use different software and hardware resources to achieve a parallel implementation of the best sequential method, that we have proposed in this paper, GenMinGen method. Such implementation requires a complex design to store the previously generated minimal generators without interfering the parallel execution.

A deeper study of the best values to be considered as BOV to maximize the results will be also developed, such as taking into account another parameters beyond the implication set cardinality.

Acknowledgements The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Supercomputing and Bioinnovation Center of the University of Málaga - Andalucía Tech (SCBI), particularly to Dr. Rafael Larrosa and Dr. Darío Guerrero. We also want to mention the orientation provided by Dr. José Antonio Onieva to identify the properties of our algorithm for a better classification of its design.

References

1. Armstrong WW (1974) Dependency structures of data base relationships. In: IFIP Congress, pp 580–583
2. Benito-Picazo F, Cordero P, Enciso M, Mora A (2017) Reducing the search space by closure and simplification paradigms. J Supercomput 73(1):75–87

3. Buchi JR, Siefkes D (1990) Finite automata, their algebras and grammars. Springer, New York Inc, Secaucus
4. Codd EF (1970) A relational model of data for large shared data banks. Commun ACM 13(6):377–387
5. Cohen E, Datar M, Fujiwara S, Gionis A, Indyk P, Motwani R, Ullman JD, Yang C (2001) Finding interesting associations without support pruning. IEEE Trans Knowl Data Eng 13(1):64–78
6. Cordero P, Enciso M, Mora A, Ojeda-Aciego M (2012) Computing minimal generators from implications: a logic-guided approach. In Szathmary L, Priss U (eds) Proceedings of the Ninth International Conference on Concept Lattices and Their Applications, Fuengirola (Málaga), Spain, October 11–14, 2012, volume 972 of CEUR Workshop Proceedings, pp 187–198. CEUR-WS.org
7. Cordero P, Mora A, Enciso M, de Guzmán IP (2002) SLFD logic: elimination of data redundancy in knowledge representation. Lect Notes Comput Sci 2527:141–150
8. de Moraes NRM, Dias SM, Freitas HC, Zárate LE (2016) Parallelization of the next closure algorithm for generating the minimum set of implication rules. Artif Intell Res 5(2):40–54
9. Doignon J, Falmagne J (1998) Knowledge spaces. Springer, Berlin
10. Dong GZ, Jiang CY, Pei J, Li JY, Wong L (2005) Mining succinct systems of minimal generators of formal concepts. Proc Database Syst Adv Appl 3453:175–187
11. Ganter B, Wille R (1999) Formal concept analysis: mathematical foundations. Springer, Berlin
12. Guigues JL, Duquenne V (1986) Famille minimale d'implications informatives résultant d'un tableau de données binaires. Math Sci Hum 24(95):5–18
13. Harper FM, Konstan JA (2015) The movielens datasets: history and context. ACM Trans Interact Intell Syst 5(4):19:1–19:19
14. Hu X, Wei X, Wang D, Li P (2007) A parallel algorithm to construct concept lattice. In Lei J (ed) Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2007, 24–27 August 2007, Haikou, Hainan, China, vol 2, pp 119–123. IEEE Computer Society
15. Kuznetsov SO, Obiedkov SA (2002) Comparing performance of algorithms for generating concept lattices. J Exp Theor Artif Intell 14(2–3):189–216
16. Maier D (1983) The theory of relational databases. Computer Science Press, Rockville
17. Missaoui R, Nourine L, Renaud Y (2010) An inference system for exhaustive generation of mixed and purely negative implications from purely positive ones. In: CEUR Workshop Proceedings, vol 672, pp 271–282
18. Missaoui R, Nourine L, Renaud Y (2012) Computing implications with negation from a formal context. Fundam Inf 115(4):357–375
19. Mora A, Enciso M, Cordero P, Fortes I (2012) Closure via functional dependence simplification. Int J Comput Math 89(4):510–526
20. Nishio N, Mutoh A, Inuzuka N (2012) On computing minimal generators in multi-relational data mining with respect to 0-subsumption. In: CEUR Workshop Proceedings, vol 975, pp 50–55
21. Poelmans J, Ignatov DI, Kuznetsov SO, Dedene G (2013) Formal concept analysis in knowledge processing: a survey on applications. Expert Syst Appl 40(16):6538–6560
22. Qu K, Zhai Y, Liang J, Chen M (2007) Study of decision implications based on formal concept analysis. Int J Gen Syst 36(2):147–156
23. Rodríguez-Lorenzo E, Cordero P, Enciso M, Mora Á (2017) Canonical dichotomous direct bases. Inf Sci 376:39–53

Capítulo 5

Sistemas de Recomendación Conversacionales

Unas veces nacen los obstáculos de la diversidad de las condiciones.

Sueño de una noche de verano

W. Shakespeare

Título:	Enhancing the conversational process by using a logical closure operator in phenotypes implications
Autores:	Fernando Benito-Picazo, Manuel Enciso, Carlos Rossi, Antonio Guevara
Revista:	Mathematical Methods in the Applied Sciences, John Wiley & Sons Ltd.
Factor Impacto JCR:	1,180. Posición 91 de 252 (Q2)
Año:	2017
Categoría:	Mathematics, Applied
Publicación:	16 febrero 2017
DOI:	10.1002/mma.4338

Special Issue Paper

Mathematical Methods in the Applied Sciences

Received 14 July 2016

Published online in Wiley Online Library

(wileyonlinelibrary.com) DOI: 10.1002/mma.4338
MOS subject classification: 68T35; 68T27; 68T30

Enhancing the conversational process by using a logical closure operator in phenotypes implications

Fernando Benito-Picazo*†, Manuel Enciso, Carlos Rossi and Antonio Guevara

Communicated by J. Vigo-Aguiar

In this paper, we present a novel strategy to face the problem of dimensionality within datasets involved in conversational and feature selection systems. We base our work on a sound and complete logic along with an efficient attribute closure method to manage implications. All of them together allow us to reduce the overload of information we encounter when dealing with these kind of systems. An experiment carried out over a dataset containing real information comes to expose the benefits of our design. Copyright © 2017 John Wiley & Sons, Ltd.

Keywords: conversational systems; feature selection; implications; logic

1. Introduction

A common problem related to knowledge discovering within the clinical context appears when it is necessary to work over datasets with a high number of features (variables or attributes). This situation is known as the curse of dimensionality phenomenon. In these cases, trying to apply data mining techniques in its different approaches (classification, regression, clustering, association rule analysis, etc.) becomes a hard task.

To address this issue, we can find many works in the literature about data reduction, specially in feature selection, that can help us to discard those features not worthy to be considered by means of different criteria. As this respect, several techniques have already been applied such as genetic algorithms, regression, neural networks and many others. All of them guided towards the application of an automated process that is applied at once (batch mode) by feature selection.

The problem of managing large volumes of information is highly present on another hot topic field of knowledge: recommender systems. The major goal of these systems is to help the user when dealing with an extremely high number of alternatives. Recommender systems are present in many different areas of today's society (e-commerce, tourism, films, music, news, etc.) in which large amount of data are pretty much often. Most recommender systems base the retrieval of items in predictions about how suitable is an item to satisfy a user's need. These predictions could be performed from user's preferences, profile, context and so on. To achieve that, different strategies are applied to enclose a recommender system into different types. Best known are content-based, knowledge-based, collaborative filtering and context-based. From the point of view of this work, we centre our efforts not just in the recommendation strategy but also in the process of obtaining a recommendation.

As is usually the case, in order to properly make a selection of items within recommenders, the user needs to introduce information over and over. That used to be something of a chore because many items may be too much detailed drawing the high-dimensionality problem mentioned earlier. An example of this situation appears when medical professionals try to elaborate a diagnosis checking symptoms from a vast list of possible ones.

One solution to tackle this issue in the field of recommender systems is the trend of conversational systems (critiquing recommender systems overall). In these systems, an iterative process is applied in which the user sets one or more features for items to fit. From this input and using different techniques, the system progresses choosing (or even predicting) subsets of items that agree with the user's preferences, until it comes to a final output with a suitable size. The problem is that, if the dataset presents a high dimensionality, the number of steps until we acquire a fitting recommendation set could be huge.

Universidad de Málaga, Andalucía Tech, Málaga, Spain
** Correspondence to: Fernando Benito-Picazo, Universidad de Málaga, Andalucía Tech, Málaga, Spain.*
† E-mail: fbenito@lcc.uma.es

The solution we proposed in this work is to manage the problem of the high dimensionality by means of a feature selection process guided by the user (human expert) within a conversational system. To achieve that, we base the approach on a novel management of implications and closure implementation. Our proposal avoids the problem mentioned in [1] remarking that batch mode feature selection systems extract them randomly. Their only criterion in the selection is the feature predictive capacity, without considering the medical knowledge. For instance, the Computer Feature Selection over Cleveland dataset discards some key features such as cholesterol, age or ECG characteristics. On the contrary, we surpass this issue by making an expert-driven automated selection.

Our major goal is to enhance the diagnosis process preserving the accuracy of the results (that is granted when using implications) and accelerating the process by reducing the necessary steps in the dialogue. Besides, the reduction of the complexity within the process is also a must, so we can obtain results timely and in due form.

As already mentioned, this work is based on implication management. Nonetheless, it is necessary to clarify that it is out of the scope of this work to describe how to extract these implications, because there are already data mining techniques specialised on this task [2].

The set of implications is the heart of the knowledge in the process we propose. We support our approach on a sound and complete logic named simplification logic (SL_{FD}), presented in several previous works, which was designed to develop deduction methods. In particular, we propose to use SL_{FD} attribute closure algorithm as the core of a feature selection framework. This algorithm is used as a basis for reasoning over the features (symptoms, phenotypes, signs) of the items (diseases) of the dataset we are going to use, reducing the number of steps in the conversation to reach a suitable diagnosis.

The key point of our framework comes from the SL_{FD} closure nature that renders a set of features that corresponds with the closure and, in addition, a new implication set corresponding with the knowledge not already used in the conversation (selection o diagnosis) process. Such implication set can be obtained by using other methods but, with SL_{FD} closure method, we compute it in linear time. This low complexity along with the reduction of the number of steps provide an overall time reduction for the process.

Our framework has been tested on a dataset that puts together real information about diseases and phenotypes. Particularly, we have selected a set of haematological diseases with phenotypes related to them. In the experiment, several metrics to evaluate the system's performance have been applied, and results have come to demonstrate the highlights of our framework.

The remainder of the paper is organised as follows: Section 2 draws attention about the state-of-the-art references and their motivations along with a brief introduction of the main elements of our approach. Section 3 brings us a detailed explanation about how the information will be managed, the notions of the SL_{FD} that apply in this approach and the specification and the usefulness of the closure algorithm. The conversational processes our approach carries out along with major benefits it reaches are presented in Section 4. Section 5 presents an empirical test considering a real case in a medical environment and the metrics used to evaluate the performance. Conclusions and future works close the paper.

2. Related works

Our approach tries to solve the problem of searching information in high-dimensionality datasets by combining techniques from diverse fields (recommender systems, feature selection, logic) in a knowledge-based framework. In this section, we analyse some outstanding related works in each area.

As mentioned, in general terms, the problem we aim to solve is to search a precise result (e.g. a diagnosis) from a high volume of data (e.g. a diseases dataset) without previous user information. By these means, our approach is inspired in one of the solutions proposed in this field, the so-called conversational recommender systems [3]. These are closely related with the concepts of critiquing recommender systems [4] and information recommendation [5]. In these systems, recommendation is generated by means of a dialogue with the user that allows an incremental elicitation of preferred item features, that is, user requirements are directly elicited within a recommendation session. An interesting work in this area is [6], which states the suitability of knowledge-based approaches for conversational processes. In particular, these authors use constraint-based reasoning, instead of our logic-based approach. Besides, this work deals with the concept of query tightening, analogous to the one applied in our proposal.

Another remarkable work is [5], which shares our aim to decrease the conversation length (number of steps). The theoretical framework they use is a model of attribute dominance with two versions, a qualitative and a quantitative one. These authors propose metrics about conversation number of steps and pruning rates, both of them very similar to the ones used in our work. Nevertheless, they present an experiment focused on tourism.

Regarding critiquing recommender systems, Chen and Pu [7] explain how user self-motivated attribute selection enable users to achieve a higher level of decision accuracy than system-proposed attribute selection. This fact supports our approach in which human expert guides the conversation and the feature selection process.

The main elements of our theoretical framework are SL_{FD} as well as the concept of attribute closure and the algorithm we have defined to compute it. One of the main advantages of this algorithm is the reduction of the time required by the selection process. For a better understanding, works related with the theoretical basis are exposed in the next section.

The starting point of our selection process is the set of implications (also named exact association rules) derived from the working dataset. To acquuire these implications, association rule mining is needed [8]. It is easy to find in literature works evaluating association rule mining techniques both in the biological–medical [9] and recommender systems [10] fields.

Once association rules had been inferred from the dataset, they can be used in the main medical tasks, such as screening, diagnosis, treatment, prognosis, monitoring and management [11, 12]. For example, Nahar *et al.* [13] uses association rule mining to analyse factors related with heart diseases. In the same way, many other works use association rules to build a decision support system or prediction model [14–17].

Closer to our work, Mansing *et al.* [18] deals with the fact that the number of association rules used to be very high. This work proposes an association rule pruning mechanism, based on well-known concepts as support, confidence and reliability. A similar idea is proposed in [19] and [20], although the latter applies temporal abstraction in the decision support system. Our approach includes an association rule filtering process based on the closure concept. Hu *et al.* [21] proposes the use of association rules for predicting combination of alarms generated by bedside monitors. These authors identify frequent alarm combinations and then carry out a variance analysis to measure the data mining process performance. Regarding our work, it should be noticed that they use a closure-like concept to define a heuristic that controls combination size. In any case, we must remark that our work is not strictly comparable with the previously cited ones, because we do not aim to build a prediction model but to improve a conversational process.

In the introduction, we affirm that our approach may be described as an expert-driven feature selection process. As Fang *et al.* [22] exposes, feature selection is a problem profusely studied in the literature. So, there are a number of solutions based on statistical techniques such as principal component analysis, linear discriminant analysis and independent component analysis, all of them suitable to deal with high-dimensionality datasets. Besides, other authors use evolutionary computing techniques such as particle swarm optimisation [23, 24]. Nevertheless, it should be remarked that most of feature selection works are focused on batch (non-interactive) processes. This way, the system, without user intervention, determines and selects more relevant features according to their predictive significance.

Recently, some papers dealing with iterative feature selection processes have been published. For example, Fialho *et al.* [25] proposes a tree feature selection process based on fuzzy modelling (and fuzzy rules). They use two tree searching techniques (sequential forward selection and sequential backward elimination). This interesting work achieves good results in well-known metrics such as Area Under Curve (AUC) or accuracy, but requires many inputs when the dataset has a high number of features. Analogously, Shilaskar and Ghatol [26] apply the same techniques but using Support Vector Machine (SVM) as classifier.

Some authors analyse the relation between features as means to reduce the dataset, for example, using correlation feature selection as a basis. This is the case of [27], although they propose a batch, non-interactive, process.

Closer to an interactive feature selection, Li *et al.* [28] define a selection process in which the input is a feature stream. They use information theory techniques (mutual information, conditional mutual information, entropy) to build a prediction model and achieve good results about the number of selected features. Although they process the features sequentially, the work is not oriented to perform as a conversational selection process.

In this way, we must remark [29], which highlights the importance of an online feature selection, so that features are processed one-by-one. This work is aimed to an extremely high-dimensionality context (in the order of millions of features) and is based on information theory. They use pairwise correlation analysis as a mean to remove redundant features. The goal of this proposal is to build a prediction mode with scalability as its main advantage.

In [30], an interesting work is presented that, as ours, deals with association rules and feature selection. Nevertheless, they base the feature extraction on methods such as partial least squares or principal component analysis, and it is not directly comparable with our approach.

3. Knowledge representation and automated deduction

In this section, we address the issue of specifying and efficiently managing the information. Knowledge-based systems have to balance both commitments to obtain, at the same time, powerful and efficient techniques. In our opinion, one of the tools that has shown a better behaviour is implications. They combine a very simple and natural way to write if-then-rules with an efficient and automated management. One evidence supporting our choice is its widespread use in different areas: databases, logic programming, formal concept analysis, artificial intelligence and so on.

In this work, knowledge is stored by considering implications, following the interpretation adopted in formal concept analysis [31], because of their simplicity. In this area, implications are inferred from datasets that are considered as binary relations between a finite set of objects and their attributes (depicted by rows and columns, respectively). Such interpretation is the following: Given a formal context K, an implication is an expression $A \rightarrow B$, where A and B are subsets of attributes, and it is said to be valid in K if and only if every object that has all the attributes from A also has all the attributes from B. Example 1 illustrates the knowledge captured by using implications.

Example 1

Let K be the formal context described in Table I showing the 22 common viruses and their usual ways of transmission.

The implications that hold in this dataset are as follows:

`Blood` → `Sexual`	`Droplet` → `Direct`
`Fluids` → `Vertical`	`Respiratory` → `Direct,Droplet`
`Direct,Sexual,Droplet` → `Faecal`	`Direct,Vertical` → `Sexual`
`Faecal,Sexual` → `Direct,Droplet`	`Faecal,Direct` → `Droplet`

In this way, implications allow to express a strong relation between two subsets of attributes of our system. Moreover, such information can be interpreted in a natural way. For instance, the last implication tells us that if a virus is transmitted by a direct contact and with faecal transmission, we also have to be vigilant about the sneezes.

Our proposal to integrate implications into the conversational issue is based on the SL_{FD} [32], which constitutes a sound and complete logic. As we shall see, such a strong basis allows us to include a reasoning method in the dialogue process. As mentioned, we built our framework on implications, which constitutes the main element of SL_{FD} language. It is formally defined as follows:

Table I. Viruses and usual way of transmissions dataset.

	Faecal	Direct	Vertical	Sexual	Respiratory	Saliva	Fluids	Droplet	Blood
Adenovirus	×	×		×				×	
Coxsackievirus	×	×			×			×	
Epstein-Barr						×			
Hepatitis A	×								
Hepatitis B			×	×			×		
Hepatitis C				×					×
Herpes type 1		×				×			
Herpes type 2			×	×					
Cytomegalovirus			×				×		
Herpesvirus type 8				×		×			
HIV			×	×					
Influenza		×						×	×
Measles virus		×						×	
Mumps virus		×						×	
Papillomavirus		×	×	×					
Parainfluenza		×						×	
Poliovirus	×								
Rabies		×						×	
Respiratory syncytial		×						×	
Rubella		×			×			×	
Varicella zoster		×						×	

Definition 1
Let M be a finite set, the formulae of SL_{FD} are expressions, named implications, of the form $X \to Y$, where X and Y are subsets of M.

From now on, we use lower case letters to denote the elements in M while uppercase letters denote its subsets. We use the standard notation and symbols of set theory. For the sake of readability, inside of a formula, $X - Y$ denotes the set difference operator $X \setminus Y$, and XY denotes the union operator $X \cup Y$.

Implications are interpreted in a conjunctive way, that is, they correspond to formulas $a_1 \wedge \ldots \wedge a_n \to b_1 \wedge \ldots \wedge b_m$ where propositions $a_1, \ldots a_n, b_1, \ldots, b_m$ are elements of the set M. The interpretation is the following:

Definition 2
Let O and M be two finite sets, named objects and attributes, respectively, and I a relation in $O \times M$. An implication of SL_{FD} $X \to Y$, where X and Y are subsets of M, is valid in I if and only if

$$\{o \in O \mid (o, x_i) \in I \, \forall x_i \in X\} \subseteq \{o \in O \mid (o, y_j) \in I \, \forall y_j \in Y\}$$

Apart from its natural way to express knowledge as a rule, implications provide a logic reasoning and inference. Their symbolic management was originally proposed in [33]. However, because of the central role that transitivity plays in that axiomatic system, the development of executable methods to solve implications problems has rest on indirect methods. The introduction of the SL_{FD} opened the door to the development of automated reasoning methods directly based on its novel axiomatic system [34, 35]. The axiomatic system of SL_{FD} is introduced as follows:

Definition 3
The axiomatic system of SL_{FD} considers reflexivity as axiom scheme

$$[\texttt{Ref}] \quad \frac{}{A \to A}$$

together with the following inference rules called fragmentation, composition and simplification, respectively.

$$[\texttt{Frag}] \frac{A \to BC}{A \to B} \qquad [\texttt{Comp}] \frac{A \to B, \, C \to D}{AC \to BD} \qquad [\texttt{Simp}] \frac{A \to B, \, C \to D}{A(C - B) \to D}$$

As we mentioned earlier, in [32], we defined, in the usual way, the semantic entailment ($\Gamma \models A \to B$) and syntactic derivation ($\Gamma \vdash A \to B$). Because we also proved the soundness and completeness of this logic, both notions can be equivalently used. We also introduced the notion of equivalence between sets of implications $\Gamma_1 \equiv \Gamma_2$ iff for all $A \to B \in \Gamma_1$, we have that $\Gamma_2 \vdash A \to B$ and vice versa.

We remark that SL_{FD} language considers as valid formulae those ones where any of their two parts can be the empty set, denoted $A \to \top$ and $\top \to A$. Their meanings were discussed in [32]. In that work, we also introduced the following result where the derivation of an implication $A \to B$ is reduced to the derivation of the formula $\top \to B$ having $\top \to A$. This result will be used later in the design of our novel closure method.

Proposition 3.1
For any Γ and for all $X, Y \subseteq M$, $\Gamma \vdash X \to Y$ if and only if $\Gamma \cup \{\top \to X\} \vdash \top \to Y$

The syntactic derivation provides an automated management of implications. In particular, it can be used to solve the so-called implication problem: Given a set of implications Γ and an implication $A \to B$, we want to answer whether $A \to B$ is deduced from Γ. This problem can be approached by using the closure operator.

Definition 4
Let $A \subseteq M$ be a set of attributes and Γ a set of implications; we define its closure with respect to Γ as the maximum subset $A^+_\Gamma \subseteq M$ such that the $\Gamma \vdash A \to A^+_\Gamma$.
Moreover, the set A is named closed iff we have $A^+_\Gamma = A$.

Implication problem has been traditionally tackled by using a basic method that receives $A \subseteq M$ as input and exhaustively uses the subset relation by iteratively traversing Γ and adding new elements to the closure. This method was proposed in the 1970s [36], and it is sketched in Algorithm 1.

Algorithm 1: Standard Closure

```
Data: Γ, A
Result: A+_Γ
1  begin
2      A+_Γ := A
3      repeat
4          A' := A+_Γ
5          foreach X → Y ∈ Γ do
6              if X ⊆ A+_Γ and Y ⊄ A+_Γ then
7                  A+_Γ := A+_Γ ∪ {Y}
8      until A+_Γ = A';
9      return A+_Γ
```

Later, several authors have developed several methods by using different techniques, efficiently solving this problem in linear time. In [37], the authors show that the complexity of closure problem is $O(|\mathcal{A}|\,|\Gamma|)$. They also mention that 'in the literature, $O(|\mathcal{A}|\,|\Gamma|)$ is usually considered as the order of the input. From this point of view, this is a linear time complexity for the computation of the closure of a set of attributes'.

In [38], we presented an attribute closure method closely tied to the SL_{FD} axiomatic system. We also showed that our method has a better performance than those based on classical closure. In this paper, we are going to use this method taking advantage of its novel characteristic.

Apart from having a strong base and good performance, one innovative feature of our method is that its output is twofold: besides the A^+_Γ set constituting the closure of the input attribute set A, it also renders a reduced set of implications that encloses the semantics that is outside the set A^+_Γ. We would like to remark that these two inputs are computed in linear time, because the subset of reduced implications is computed by the algorithm at the same time it computes the attribute closure.

The kernel of this closure method is the existence of three equivalences that can be enunciated by using SL_{FD}:

- Equivalence I: If $U \subseteq W$, then $\{\top \to W, U \to V\} \equiv \{\top \to WV\}$
- Equivalence II: If $V \subseteq W$, then $\{\top \to W, U \to V\} \equiv \{\top \to W\}$
- Equivalence III: If $U \cap W \neq \varnothing$ or $V \cap W \neq \varnothing$, then $\{\top \to W, U \to V\} \equiv \{\top \to W, U - W \to V - W\}$

The closure method works as follows. To compute the attribute closure of A with respect to Γ, the method is triggered by the seed formula $\top \to A$ and, exhaustively executing these three equivalences, it modifies this implication rendering $\top \to A^+_\Gamma$ and, at the same time, a reduced set of implications Γ'. The method is described in Algorithm 2.

We end this section with an illustrative example.

Example 2
Let Γ be the set of implications from Example 1. We show how SL_{FD} closure computes the closure of the attribute `Droplet`. In Table II, we show the application of the equivalences to each implication in the Γ set and how the set of attributes grows. We would like to remark that, in this example, only one repeat loop is needed.

As a final conclusion of this example, our method received the set Γ and `Droplet` attribute and renders as output the pair: ⟨{`Droplet, Direct`}, {`Blood` → `Sexual`; `Fluids` → `Vertical`; `Sexual` → `Faecal`; `Vertical` → `Sexual`}⟩. That is to say, `Droplet`$^+$ and the reduced set of implications that stores the knowledge complementing the closure set.

Algorithm 2: The SL_{FD} Closure

Data: Γ, X
Output: $\langle X^+, \Gamma' \rangle$
begin
  $\Delta := \langle X, \Gamma \rangle$
  repeat
    $\Gamma' := \Gamma$
    foreach $Y \to Z \in \Gamma$ **do**
      if $Y \subseteq X$ **then** /* Equivalence I */
        $\Delta := \langle XZ, \Gamma \setminus \{Y \to Z\} \rangle$
      else if $Z \subseteq X$ **then** /* Equivalence II */
        $\Delta := \langle X, \Gamma \setminus \{Y \to Z\} \rangle$
      else if $Y \cap X \neq \varnothing$ **or** $Z \cap X \neq \varnothing$ **then** /* Equivalence III */
        $\Delta := \langle X, (\Gamma \setminus \{Y \to Z\}) \cup \{(Y - X) \to (Z - X)\} \rangle$
  until $\Gamma' = \Gamma$;
  return Δ

Table II. An application example of SL_{FD} closure.

Closure	Implication	New implication	
Droplet			
Droplet	Blood → Sexual	Blood → Sexual	-
Droplet, Direct	Droplet → Direct	×	Equiv. I
Droplet, Direct	Fluids → Vertical	Fluids → Vertical	-
Droplet, Direct	Respiratory → Direct, Droplet	×	Equiv. II
Droplet, Direct	Direct, Sexual, Droplet → Faecal	Sexual → Faecal	Equiv. III
Droplet, Direct	Direct, Vertical → Sexual	Vertical → Sexual	Equiv. III
Droplet, Direct	Faecal, Sexual → Direct, Droplet	×	Equiv. II
Droplet, Direct	Faecal, Direct → Droplet	×	Equiv. III

SL_{FD}, simplification logic.

4. Conversational process

Once we have presented our basis, in this section, we depict a detailed explanation of the dialogue process system along with a basic schema to facilitate the comprehension.

First of all, we depart from the premise that we have a dataset containing diseases and phenotypes and the set of implications that holds on it. This is considered the starting point at which this work begins and, as mentioned before, it stays out of the scope of this work. From there on, the dialogue process will go along through the following points:

(1) Once we count on this information, the user starts interacting with the system by selecting a phenotype she suspects a disease to be related to. In this paper and with the intention of illustration, we are limiting the number of phenotypes to be selected to one at a time in each step of the conversation. This way, we can easily appreciate how the system performs. Yet a generalisation system considering more than one phenotype per step showed a similar behaviour but going faster.
(2) Then, the process flows into the closure algorithm calculating both the phenotype's closure set for the selected one and also the reduced set of implications that corresponds with the complement of this closure.
(3) Once the closure algorithm has finished, a first possible diagnosis is shown. This diagnosis shows the list of diseases in the dataset agreeing with the selected phenotypes. In order to acquire this list, the system launches a query to the database requesting those diseases that verify the selected phenotypes.
(4) At this point, the user can stop the dialogue in the case that she is already satisfied with the result (a list of diseases), or she can go ahead trying to acquire a more reduced diagnosis.
(5) Finally, the user can return to step 2 selecting new phenotypes until she acquires a satisfying set of disease result (possible diagnosis) or the system runs out of phenotypes.

The main contribution of this paper is the benefits of reducing the number of available phenotypes (in general, the attributes space), clarifying in each step the user scene. For further steps in the dialogue, we reduce the number of available phenotypes deleting those included in the closure set. As a consequence, some closure's phenotypes remain hidden to the user, in an intelligent and consistent way, because they are already implicitly included. This saves the user unnecessary efforts about information overload. However, even that this could be accomplished by classic closure algorithms, the major novelty of our method is that, *at the same time*, we also reduce

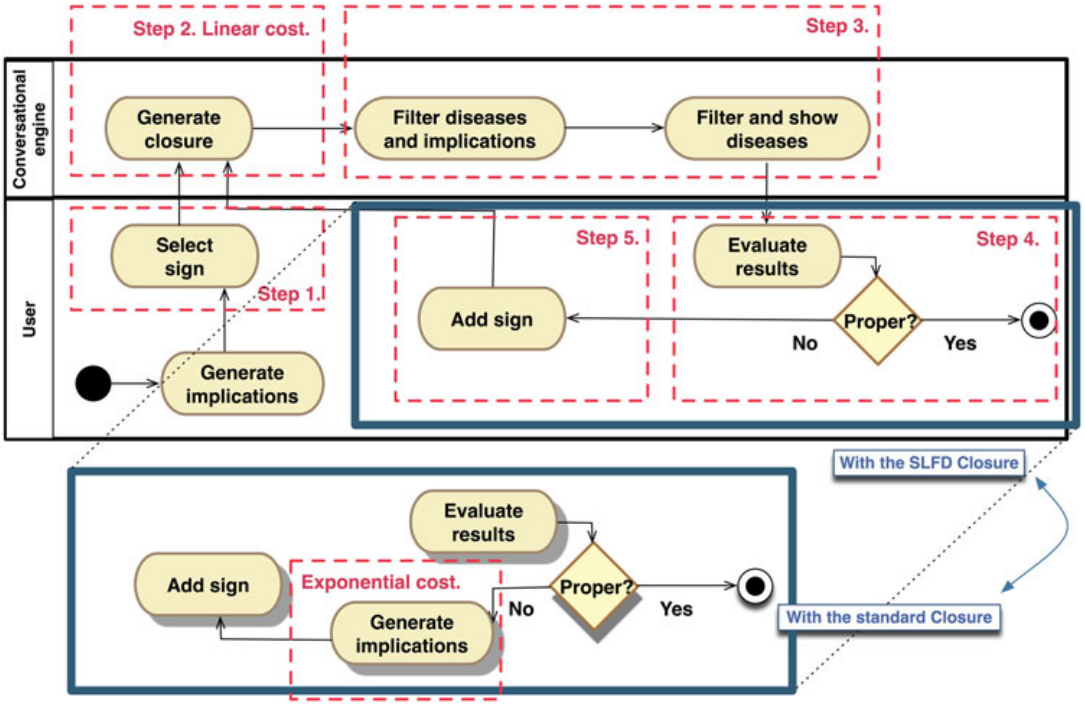

Figure 1. Dialogue process schema using simplification logic (SL_{FD}) closure versus standard closure.

the number of implications. This fact totally places us in a privileged position, overtaking the hard cost of a data mining process to extract the new set of implications for the reduced dataset after each searching step, as necessary when using classical implementations. On the contrary, in every refining attempt in the dialogue using our approach, we do not need to start the process from the beginning but continuing from there, where both symptoms and implications have been decreased. Consequently, the process overcomes the data mining costs preserving a linear complexity along the dialogue, and the interaction becomes truly faster. Figure 1 depicts the aforementioned steps by using the activity diagram of the UML language.

As mentioned earlier, in step 0, we need to count on the inferred set of implications within the dataset (generate implications activity). Although this is an exponential task, it is independent of the closure strategy we are willing to use. Thereafter, we execute step 1 and select the observed sign to continue with the execution of the closure method itself (generate closure activity). Observe that either our closure implementation or classical ones are able to perform in linear time. Now, however, it is remarkable that no matter how many steps we want the dialogue to go further, there is no need of mining any new set of implications; we already have it each time our closure implementation is applied because it is narrowing both attributes and implications at the same time thanks to the output of SL_{FD} closure (filter diseases and implications activity).

Nevertheless, in the case of classical closure implementations, the disadvantage arises as follows. Because only attributes are narrowed each time the closure applies, in order to proceed with the next step of the conversation, we need to generate a new set of implications accordingly to the new narrowed set of attributes. Therefore, this is an exponential task that is mandatory after every step of the dialogue. As a consequence, the overall complexity of the process becomes exponential.

Once the advantages of our method have been exposed, we proceed now with an example that illustrates the achievement of our approach about avoiding the exponential complexity of the conversational process.

Example 3

In the main experiment of this paper (which will be explained in Section 5.2), we have used a real dataset matching information about diseases and phenotypes. From the different runs of this experiment, five steps have been the maximal length the dialogue reached in a simulated conversation. Therefore, we are going to compare here this dialogue with such limit situation in two possible scenarios, the use of SL_{FD} closure and the use of any other classical closure implementations regarding the implication mining task. Here, we do not worry about the execution time of the closure method itself, because it is common to both approaches and not relevant compared with the implication mining cost.

This comparison is depicted in Table III. This table counts on five columns. First one indicates the number of steps along the conversation. Second one denotes the phenotype selected at each step of the dialogue. Columns 3 and 4 are the most important ones; they show the time needed to enumerate the new set of implications after each new step using classical closure implementations (column 3) and SL_{FD} closure (column 4). Last column shows the cardinal of the regenerated set of implications. At length, we show the time needed in the process using either a classical closure implementation or SL_{FD}.

Table III. Total time saved enumerating the set of implications by using SL_{FD} closure.

Step	Phenotype selected	Standard closure	SL_{FD} closure	Number of implications
1	HPO_7	22 s 181 ms	—	8.578
2	HPO_639	21 s 526 ms	—	8.270
3	HPO_2910	20 s 324 ms	—	7.988
4	HPO_1250	19 s 627 ms	—	7.534
5	Dialogue ends	—	—	—
	Total time saved	83 s 658 ms	—	

SL_{FD}, simplification logic; HPO, Human Phenotype Ontology Consortium.

These measures have been obtained by virtue of the application of the specific package for R language named *Arules: Mining Association Rules and Frequent Itemsets*[‡]. Moreover, the hardware configuration used goes as follows: Intel Core 2 Duo 2.6 Ghz, 4 Gb RAM running over Windows 7.

The significant overall reduction of time obtained by using SL_{FD} closure is because of the linear calculation of the new set of implications to be used in the next step, whereas classical implementations suffer from regenerating implications again and again in each step.

5. Application of SL_{FD} conversational method to hematologic diseases selection

In this section, we describe the promising results obtained by our method in a real case. First, we establish the metrics defined to measure the benefits of our method and, later, we describe the selected dataset and the results of the experiment over it.

5.1. *Evaluation metrics*

When trying to enhance the interaction within a conversational system, evaluating the length of the dialogue for a typical query could be considered the most basic test [39]. Another popular measures to evaluate the effectiveness of recommender systems point to *precision* and *recall* measures [40]. Precision, defined as P = TP/(TP + FP), where TP means the number of true positives, FP the number of false positives and FN the number of false negatives, determines the fraction of relevant items retrieved out of all items retrieved. On the other hand, recall R = TP/(TP + FN), which determines the fraction of relevant items retrieved out of all relevant items. These two popular measures may not shed light on the matter of evaluating this approach, and the reason is twofold. First, every disease belonging to the list of resulting diagnoses agrees with the symptoms selected by the user as the database queries launched to retrieve diseases reflect these constraints. And second, after every loop of the process, we retrieve all the existing diseases in the dataset that hold with the symptoms selected,that is, all the relevant elements. Something similar occurs with other historical measures like mean absolute error or root mean square error. These two measures based on ratings have no place in this approach because no ratings are considered in order to conduct the conversation and the final items retrieval. In addition, there is no need to explicitly consider an accuracy measure because we do not build a prediction model, instead the use of implications ensures full accurate results. Fortunately, there are also others measures that could fairly reflect the statistics of the experiments. We continue describing those ones considered in this work.

5.1.1. Number of steps (N). This metric evaluates the actual length in steps of the conversation. It is interesting in the sense that it offers a rapid overview of the length of the interaction between the user and the conversational system. That gives an idea of whether the conversation has quickly satisfied the user or too many steps were needed. At this regard, within the following experiments, we will consider, without loss of generality, the user satisfied when a result of five (or less) diseases is returned. We set this limit because it deems advisable in order to constitute a proper diagnosis. Simultaneously, it represents the number of attributes (phenotypes) requested by the user because we are hitherto selecting one attribute at a time.

$$N = |\text{Selected attributes}|, \quad \text{where } |A| \text{ represents the cardinal of A.}$$

5.1.2. Speed of pruning at step i (S_i). This metric evaluates the percentage of the attributes the user is saving over the course of the conversation, accumulating from one step to another. When using this metric, we are willing to noticed whether the pruning rates have been better at the first steps of the conversation or at the last ones. Overall, it is a metric to measure how *fast* the system removes the overload of information. Notice that as mentioned before, we are taking one attribute at a time in this approach.

$$S_i = \frac{|\text{Attribute Closure}|_i - i}{|\text{M}|}, \quad i = 1, \ldots, N, \text{ and M represents the whole set of attributes as shown in Section 3.}$$

5.1.3. Attributes pruning (P). This last metric is equal to the speed of pruning but taking values at the end of the dialogue. It represents the percentage of the attributes that have been removed from the original set throughout the conversation. The pruning of these attributes is consequent because they are implicit by the user's selected ones. Formally,

$$P = S_N$$

Once all the basis have been defined and the metrics are explained and formulated, we are ready to begin the experiments.

[‡] *https://cran.r-project.org/web/packages/arules/index.html.*

Table IV. Diseases and symptoms dataset (extract).

Disease ID	HPO_1249	HPO_1250	HPO_1251	HPO_1252	HPO_1254	HPO_1257	…
274000		X					
275630	X		X				
277380				X	X		
300884		X		X			
300322	X			X		X	
…							

HPO, Human Phenotype Ontology Consortium.

5.2. Experiments and results

In this section, we are going to perform experiments on a real-world dataset with a substantial amount of information. First of all, we are going to provide information about the dataset we are going to work with.

The source from which we have extracted the data is the Human Phenotype Ontology Consortium[§] (HPO). As can be read in their web page: *'HPO* [41] *aims to provide a standardized vocabulary of phenotypic abnormalities encountered in human disease. Each term in the HPO describes a phenotypic abnormality. The HPO is currently being developed using the medical literature, Orphanet*[¶]*, DECIPHER*[‖]*, and OMIM*[**]*. The HPO is developed within the context of the Monarch Initiative*[††]*'.*

From this information, we have been able to generate a dataset on which we shall perform our experiments. Because the amount of information of HPO is huge, in this first approach, we are just going to use an extract of all the information available. In combination with Online Mendelian Inheritance in Man (OMIM) to distinguish amidst different types of diseases present in HPO databases, we have generated a table matching haematologic diseases and phenotypes,because the resulting information is substantial. Table IV shows an extract of the whole generated table. In the case we wish to obtain a detailed explanation of every disease shown forward, we commend the reader to visit OMIM web page and feed its search engine with the identifiers listed in Disease ID column. As an example, Disease ID 275630 corresponds to Chanarin-Dorfman syndrome.

Now, once we have presented our dataset, the next stage goes in using one of the previously mentioned techniques to retrieve all the implications that hold on it. Summing up, we are going to work over a dataset with 446 diseases, 100 different phenotypes and the set of implications that holds on it. Unfortunately, we cannot show all these implications here because of the obvious space limitations because the implications set goes beyond 6.000 implications.

In this point, we need to make an aside because from a purist view, if we calculate the so-called Duquenne-Guigues base [42] of implications that holds in context, there are 8.811 implications; so why have we pruned the set? Actually, the main feature of Duquenne-Guigues base of implications is that this base has a minimal possible number of implications among all possible bases of implications that hold in context. However, there will be several implications where there are no items that support (as stated in association rules theory) them, and usually such implications mean that set of items, contained in premise, does not occur together in context. Also, such implications include all attributes from context. Hence, the sense of this kind of implications is just theoretical and has nothing remarkable when dealing with real-world applications; that is why we ended up dismissing them. Nonetheless, the attributes present in these zero-supported implications are of course considered along the process and are fully accessible for the user to be selected.

That being said, the way on how we are going to proceed goes as follows. We are going to perform a test consisting of 1.000 simulations following to the letter the process depicted in Section 4. However, these runs will be carried out as random simulated dialogues. That is to say, we are going to conduct every dialogue by selecting phenotypes randomly from the set of available ones in each step. This strategy could give different readings. At a glance, it may deem advisable to go ahead with the random selection of phenotypes, so the reliability of the experiment is granted, and there is no possibility of inducing favourable situations for us to obtain better results. Yet the random strategy could also overshadow the benefits of our approach. On the one hand, imagine a situation where the random strategy falls into selecting phenotypes without any relation to each other. Then, the dialogue will end up quickly as there will be few diseases (or even no one) matching this phenotype's selection, and the process would finish with no possibility of applying any pruning at all. On the other hand, suppose a real dialogue where the phenotypes related to the patient share some sort of relation to each other. Then, during the dialogue, our process will be able to perform better pruning as the input phenotypes do have certain linkages; so, such a *fair-play* interaction would definitely highlight our approach. Needless to say that every result shown along with the experiments is the fruit of a statistical study [43] behind the results given from every run, so we kept the most reliable ones. Finally, in line with everything earlier, we are ready to launch the experiment.

At the end of the experiment, Figure 2 comes to clearly illustrate the results obtained for the number of steps metric after simulating 1.000 different conversations.

[§] http://www.human-phenotype-ontology.org.
[¶] http://www.orpha.net/consor/cgi-bin/index.php.
[‖] https://decipher.sanger.ac.uk.
[**] http://www.omim.org.
[††] https://monarchinitiative.org.

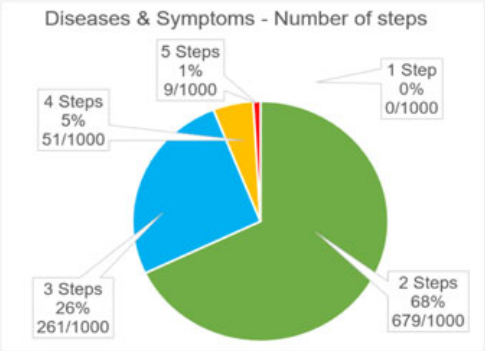

Figure 2. Number of steps metric results for the complete experiment.

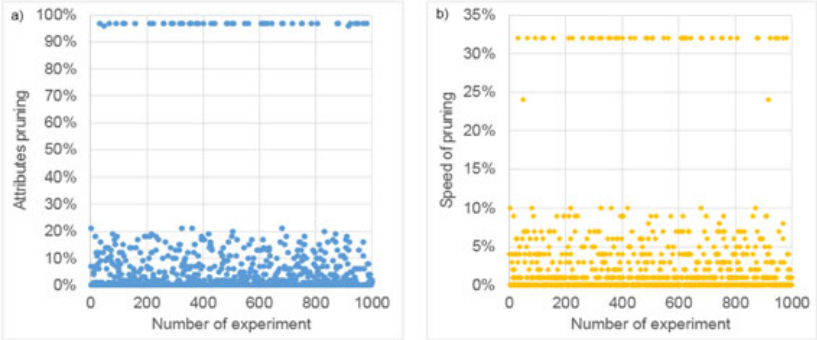

Figure 3. Attributes pruning (a) and speed of pruning (b) values for the complete experiment.

A glance is enough to easily realise that the conversational system is capable of guiding the dialogue to a suitable final diagnosis in two to three steps most of case. Keeping in mind the size of this dataset, these are encouraging results so far. There are other cases where the conversation has taken longer reaching four steps or even five, yet these cases are few and far between. Finally, only two experiments within the 1.000 runs have finished in just one step. These cases appear because the patient reveals a phenotype that appears in less than five of the diseases of the dataset and then the dialogue finishes at a glance (remember that for the random experiments, we established a limit of five or less diseases as the maximum to consider a diagnosis as a good option). Overall, we can consider that this number of step values fairly surpass previous studies [44] where the amount of input data provided is less than the one tested here.

Regarding the attributes pruning values, Figure 3(a) shows that in the vast majority of cases, the conversational system has freed us to worry about around 5–20% of attributes along the dialogue. In addition, there have been cases where the pruning did its utmost, reaching 97% of phenotypes, because the successive selected phenotypes conform a combination that brought the closure implementation to highly reduce the sets of both attributes and implications.

Speed of pruning values go hand-in-hand with the attributes pruning as it can be easily realised by matching Figure 3(a) and (b). Therefore, we can appreciate a general trend of experiments reducing attributes at an average speed of 5–10% per step and other ones achieving higher rates hovering 30–35% of attributes.

6. Conclusions and future works

We have presented here a novel application of our SL_{FD} closure algorithm in order to face the problem of the overwhelming dimensionality within the datasets. Our solution proposes a conversational process of user-driven feature selection. This work merges features of knowledge-based systems in combination with an appropriate management of implications through SL_{FD} closure. All these characteristics move us to favourably improve the diagnosis process with a high reduction in the length of the conversation and yet preserving the system's accuracy. That is not to forget the benefits we provide in terms of execution times reducing the conversational process from exponential to linear complexity.

Additionally, in the light of the results obtained over a dataset containing real information, we are of the opinion that this course of action heads the research to the right direction. The pruning rates reflect the good deeds of our approach improving the user–system interaction. Also, the number of step values encourage us to go straight on larger datasets because the actual numbers are admissible. Not forgetting, of course, the fact that results ensure 100% of accuracy.

Finally, our system constitutes a framework that can be integrated on diverse datasets and results stay admissible. This is certainly a major contribution of this work, because the possibilities it offers can reach many applications in different areas.

As future works, our results motivate a number of important directions for further research. Trying to discover which characteristics concerning the dataset (dimensionality, sparsity, etc.) are relevant to explain how the information extracted from the dataset behaves is an extended avenue for future researching tasks. In the same direction, identifying elements within the dataset, which play a more significant role amidst the others (either for being more frequent, unique, close-related to others, etc.), could help us to guide the conversation into the proper direction in a more comfortable way.

Acknowledgements

The authors thankfully acknowledge both the information provided by the Human Phenotype Ontology Consortium (HPO) and the Online Mendelian Inheritance in Man (OMIM) from the McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University School of Medicine.

This work was supported by project TIN2014-59471-P of the Science and Innovation Ministry of Spain, co-funded by the European Regional Development Fund (ERDF).

References

1. Nahar J, Imam T, Tickle KS, Chen YPP. Computational intelligence for heart disease diagnosis: a medical knowledge driven approach. *Expert Systems with Applications* 2013; **40**(1):96–104.
2. Krajca P, Outrata J, Vychodil V. Advances in algorithms based on CBO. *Proceedings of the 7th International Conference on Concept Lattices and their applications*, Sevilla, Spain, 2010, 325–337.
3. Guerrero SE, Salamo M. Increasing retrieval quality in conversational recommenders. *IEEE Transactions on Knowledge and Data Engineering* 2012; **24**(10):1876–1888.
4. Reilly J, McCarthy K, McGinty L, Smyth B. Incremental critiquing. *Knowledge-Based Systems* 2005; **18**(4–5):143–151.
5. Trabelsi W, Wilson N, Bridge D, Ricci F. Preference dominance reasoning for conversational recommender systems: a comparison between a comparative preferences and a sum of weights approach. *International Journal on Artificial Intelligence Tools* 2011; **20**(04):591–616.
6. Jannach D, Zanker M, Fuchs M. Constraint-based recommendation in tourism: a multiperspective case study. *Information Technology & Tourism* 2009; **11**(2):139–155.
7. Chen L, Pu P. Hybrid critiquing-based recommender systems. *Proceedings of the 12th International Conference on Intelligent User Interfaces*, IUI '07, ACM, New York, NY, USA, 2007, 22–31.
8. Rodríguez-Jiménez JM, Cordero P, Enciso M, Mora A. Data mining algorithms to compute mixed concepts with negative attributes: an application to breast cancer data analysis. *Mathematical Methods in the Applied Sciences* 2016:4829–4845, DOI 10.1002/mma.3814.
9. Rodriguez A, Carazo JM, Trelles O. Mining association rules from biological databases. *Journal of the American Society for Information Science and Technology* 2005:493–504, DOI 10.1002/asi.20138.
10. Smyth B, McCarthy K, Reilly J, O'Sullivan D, McGinty L, Wilson DC. Case studies in association rule mining for recommender systems. *Ic-Ai*, Las Vegas, Nevada, USA, 2005, 809–815.
11. Esfandiari N, Babavalian MR, Moghadam AME, Tabar VK. Knowledge discovery in medicine: current issue and future trend. *Expert Systems with Applications* 2014; **41**(9):4434 –4463.
12. Pandey B, Mishra RB. Knowledge and intelligent computing system in medicine. *Computers in Biology and Medicine* 2009; **39**(3):215 –230.
13. Nahar J, Imam T, Tickle KS, Phoebe Chen Y-P. Association rule mining to detect factors which contribute to heart disease in males and females. *Expert Systems With Applications* 2013; **40**:1086–1093.
14. Anooj PK. Clinical decision support system: risk level prediction of heart disease using weighted fuzzy rules. *Journal of King Saud University – Computer and Information Sciences* 2012; **24**(1):27–40. arXiv:1011.1669v3.
15. Huang MJ, Chen MY, Lee SC. Integrating data mining with case-based reasoning for chronic diseases prognosis and diagnosis. *Expert Systems with Applications* 2007; **32**(3):856–867.
16. Imberman SP, Domanski B, Thompson HW. Using dependency/association rules to find indications for computed tomography in a head trauma dataset. *Artificial Intelligence in Medicine* 2002; **26**(1):55–68.
17. Nahar J, Tickle KS, Ali AB, Chen YPP. Significant cancer prevention factor extraction: an association rule discovery approach. *Journal of Medical Systems* 2011; **35**(3):353–367.
18. Mansingh G, Osei-Bryson KM, Reichgelt H. Using ontologies to facilitate post-processing of association rules by domain experts. *Information Sciences* 2011; **181**(3):419–434.
19. Lee DG, Ryu KS, Bashir M, Bae JW, Ryu KH. Discovering medical knowledge using association rule mining in young adults with acute myocardial infarction. *Journal of medical systems* 2013; **37**(2):98–96.
20. Yeh JY, Wu TH, Tsao CW. Using data mining techniques to predict hospitalization of hemodialysis patients. *Decision Support Systems* 2010; **50**: 439–448.
21. Hu X, Sapo M, Nenov V, Barry T, Kim S, Do DH, Boyle N, Martin N. Predictive combinations of monitor alarms preceding in-hospital code blue events. *Journal of Biomedical Informatics* 2012; **45**(5):913–921.
22. Fang R, Pouyanfar S, Yang Y, Chen SC, Iyengar SS. Computational health informatics in the big data age. *ACM Computing Surveys* 2016; **49**(1):1–36.
23. Inbarani HH, Azar AT, Jothi G. Supervised hybrid feature selection based on {PSO} and rough sets for medical diagnosis. *Computer Methods and Programs in Biomedicine* 2014; **113**(1):175 –185.
24. Vieira SM, Mendonça LF, Farinha GJ, Sousa JMC. Modified binary {PSO} for feature selection using {SVM} applied to mortality prediction of septic patients. *Applied Soft Computing* 2013; **13**(8):3494 –3504.
25. Fialho AS, Cismondi F, Vieira SM, Reti SR, Sousa JMC, Finkelstein SN. Data mining using clinical physiology at discharge to predict {ICU} readmissions. *Expert Systems with Applications* 2012; **39**(18):13158–13165.

26. Shilaskar S, Ghatol A. Feature selection for medical diagnosis: evaluation for cardiovascular diseases. *Expert Systems with Applications* 2013; **40**(10):4146–4153.
27. Mathias JS, Agrawal A, Feinglass J, Cooper AJ, Baker DW, Choudhary A. Development of a 5 year life expectancy index in older adults using predictive mining of electronic health record data. *Journal of the American Medical Informatics Association* 2013; **20**(e1):e118–24.
28. Li H, Wu X, Li Z, Ding W. Online group feature selection from feature streams. *AAAI*, Bellevue, Washington, USA, 2013, 1627–1628.
29. Yu K, Wu X, Ding W, Pei J. Towards scalable and accurate online feature selection for big data. *2014 IEEE International Conference on Data Mining*, Shenzhen, China, 2014, 660–669.
30. Chaves R, Ramírez J, Górriz JM, Puntonet CG. Association rule-based feature selection method for Alzheimer's disease diagnosis. *Expert Systems with Applications* 2012; **39**(14):11766–11774.
31. Ganter B, Wille R. *Formal Concept Analysis: Mathematical Foundations* 1st ed. Springer-Verlag New York, Inc.: Secaucus, NJ, USA, 1997.
32. Cordero P, Enciso M, Mora A, de Guzmán IP. SLFD logic: elimination of data redundancy in knowledge representation. *IBERAMIA 2002: Proceedings of the 8th Ibero-American Conference on AI*, Springer-Verlag, London, UK, 2002, 141–150.
33. Armstrong WW. Dependency structures of data base relationships. *IFIP Congress*, Amsterdam, Holland, 2002, 580–583.
34. Cordero P, Enciso M, Mora A, de Guzmán IP. A tableaux-like method to infer all minimal keys. *Logic Journal of the IGPL* 2014; **22**(6):1019–1044.
35. Cordero P, Enciso M, Mora A, Ojeda-Aciego M, Rossi C. Knowledge discovery in social networks by using a logic-based treatment of implications. *Knowledge-Based System* 2015; **87**:16–25.
36. Maier D. *The Theory of Relational Databases*. Computer Science Press: Rockville, 1983.
37. Paredaens J, Bra PD, Gyssens M, Gucht DV (eds). *The structure of the relational database model*, EATCS Monographs on Theoretical Computer Science: Springer-Verlag Berlin, 1989.
38. Mora A, Cordero P, Enciso M, Fortes I, Aguilera G. Closure via functional dependence simplification. *International Journal of Computer Mathematics* 2012; **89**(4):510–526.
39. McSherry D. Minimizing dialog length in interactive case-based reasoning. *Proceedings of the 17th International Joint Conference on Artificial Intelligence, IJCAI*, Seattle, Washington, USA, 2001, 993–998.
40. Ricci F, Rokach L, Shapira B, Kantor PB (eds). *Recommender Systems Handbook*. Springer: USA, 2011.
41. Köhler S, Doelken SC, Mungall CJ, Bauer S, Firth HV, et al. The Human Phenotype Ontology project: linking molecular biology and disease through phenotype data. *Nucleic Acids Research* 2014; **42**(Database Issue):D966–D974.
42. Guigues JL, Duquenne V. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Humaines* 1986; **95**:5–18.
43. Goh TN. An information management paradigm for statistical experiments. *Quality and Reliability Engineering International* 2010; **26**(5):487–494.
44. Li H, Wu X, Li Z, Ding W. Online group feature selection from feature streams. *AAAI*, AAAI Press, Bellevue, Washington, USA, 2013.

Capítulo 6

Conclusiones y Trabajos Futuros

Si una conclusión no está poéticamente equilibrada, no puede ser científicamente cierta.

Los robots del amanecer
I. Asimov

Fundamentalmente, la naturaleza dual de esta esta tesis ha conllevado dos grandes grupos de tareas. Por un lado, se ha realizado un profundo estudio de los métodos basados en la lógica para el tratamiento eficiente de la información utilizando los conjuntos de implicaciones que se verifican en un determinado *dataset*. Y por otro lado, se han realizado una serie de tareas para contrastar la validez de estos métodos teóricos en la práctica.

Se ha investigado sobre tres problemas diferentes: claves minimales, generadores minimales y sistemas de recomendación. Para cada una de ellas se han realizado multitud de experimentos que demuestran la utilidad y la validez del trabajo realizado. Asimismo, se hace un especial hincapié en la parte aplicada del estudio con la intención de facilitar la transferencia de conocimiento a entornos diferentes del ámbito académico, como el mercado empresarial.

A lo largo de la tesis se puede apreciar el hecho de que contar con una sólida teoría basada en la Lógica y las Matemáticas concede la base

para la creación de métodos automatizados con los que poder afrontar el desarrollo de aplicaciones de ingeniería. De esta forma, se ha comprobado que existe una gran cantidad de información implícita en los datos que se suelen utilizar en dichas aplicaciones. El descubrimiento de toda esta información y su gestión inteligente es sin duda una clara oportunidad de investigación con una fuerte actividad y repercusión en la actualidad. Esta ha sido la intención principal a la hora de trabajar con FCA y los conjuntos de implicaciones. Pasar de la teoría a la práctica y viceversa ha sido uno de los principales desafíos de esta tesis al hacer que estos conceptos se conviertan en una herramienta fructífera para la representación, gestión y análisis del conocimiento en situaciones reales.

Se alcanza ahora el último capítulo de la tesis, en el cual se recopilan las conclusiones más importantes alcanzadas como resultado del trabajo de investigación realizado. Seguidamente, cerrarán el capítulo una serie de tareas con las que continuar a partir de este punto y que se presentan como trabajos futuros.

6.1. Conclusiones

En primer lugar, y dada la relación manifiesta en términos de algoritmos y computación paralela, se muestran las conclusiones referentes a claves y generadores minimales. Más adelante y para terminar la sección, se muestran aquellas conclusiones obtenidas en torno a SRs conversacionales.

Como se ha mencionado con anterioridad, conocer las claves es una tarea crucial para muchas áreas de gestión de la información. Se recuerda que una clave es un conjunto de atributos de un esquema relacional que nos permite distinguir inequívocamente cada objeto del *dataset*. El problema surge debido a la complejidad exponencial del algoritmo de búsqueda de claves a partir de un conjunto de DFs que se cumplen en un esquema del modelo relacional [74].

Para abordar este problema, a partir de los métodos que los autores presentaron en [25], se han diseñado una serie de nuevos métodos que permiten computar el conjunto de claves a partir del conjunto de implicaciones

haciendo uso de métodos de razonamiento automático basados en la lógica SL_{FD}. Se han diseñado e implementado, pasando desde la teoría a la práctica, los diferentes métodos basados en el paradigma de tableaux, y se ha verificado como, hasta donde se ha podido comprobar, los resultados obtenidos mejoran los de las aproximaciones anteriores [25] reduciendo tres aspectos fundamentales: el tiempo de cómputo, los cálculos redundantes y el tamaño del problema [13]. Además, en este trabajo se ha realizado el diseño y la implementación paralela del algoritmo original, poniendo de manifiesto como la introducción del paralelismo y los recursos de supercomputación permiten que las limitaciones que se encontraban en el trabajo original [25] hayan podido solventarse, abriendo la puerta a poder trabajar con cantidades mayores de información.

Por otro lado, enumerar todos los conjuntos cerrados y sus generadores minimales es también un problema muy complejo pero esencial en varias áreas de conocimiento, constituyendo una oportunidad para mostrar los beneficios de FCA cuando se trabaja en aplicaciones reales. Junto con los conjuntos cerrados, los generadores minimales, son esenciales para obtener una representación completa del conocimiento en FCA [96].

El punto de partida para trabajar sobre generadores minimales en esta tesis ha sido el método presentado en [28], donde se utilizó la lógica SL_{FD} como herramienta para encontrar todos los generadores minimales a partir de un conjunto de implicaciones. La propuesta que se ha realizado en esta tesis ha consistido en el diseño y la implementación de métodos que nos permitan identificar los generadores minimales como representaciones canónicas de cada conjunto cerrado para un conjunto de implicaciones.

Desafortunadamente, la dificultad que aparece al utilizar estos métodos es que la obtención de todos los conjuntos cerrados y sus respectivos generadores minimales es un problema con complejidad exponencial.

Con la intención de afrontar esta tarea, se han diseñado dos métodos de poda para mejorar el rendimiento de la enumeración de los generadores minimales. Para ello, se ha hecho un uso intensivo de la lógica SL_{FD} sobre conjuntos de implicaciones. Finalmente, se han diseñado, analizado y probado algoritmos diferentes (MinGen, MinGenPr, GenMinGen),

mostrando claramente las mejoras aportadas por cada uno. Así, se han alcanzado reducciones de más del 50 % en el número de nodos del árbol de búsqueda que construye el método original, MinGen, frente al resto, MinGenPr y GenMinGen, como se ha detallado en [14].

Asimismo, la utilización de estrategias paralelas se alza como la mejor alternativa en la resolución tanto de los problemas de claves minimales como de generadores minimales. Este hecho se debe a que cada uno de los subproblemas que se generan en la resolución de estos problemas (cada nodo del árbol del tableaux) es una instancia equivalente del problema original pero reducida, y por tanto, pueden tratarse de manera paralela asignando cada uno de ellos a un procesador diferente.

No obstante, el primer punto que es necesario aclarar es que se ha aplicado un *paralelismo* de tipo *hardware*. Es decir, ha consistido en la utilización de un conjunto de procesadores que se encargan de ir resolviendo cada uno de los subproblemas de forma simultánea. Por lo tanto, estas implementaciones no son un caso de desarrollo de código paralelo desde una visión más centrada en la programación, sino que es más acertado considerarlas como aplicaciones basadas en una estrategia *MapReduce* [35], que se ejecutan de forma paralela con la ayuda de recursos *hardware*.

Al igual que para el problema de las claves minimales, se ha desarrollado el código necesario para poder trabajar con grandes cantidades de información para identificar los generadores minimales. Para resolver los problemas de tiempo de ejecución cuando la cantidad de información de entrada sea considerable, el desarrollo se ha optimizado para ejecuciones que sean capaces de aprovechar grandes recursos computacionales, tales como los proporcionados por el Centro de Supercomputación y Bioinnovación de la Universidad de Málaga; gracias a ellos ha sido viable realizar la gran mayoría de las pruebas. Aun contando con dichos recursos, se ha llegado a la conclusión de que, para ambos casos, claves y generadores minimales, es absolutamente necesario que las implementaciones tengan en cuenta el correcto uso de los recursos de memoria; incluso para problemas pequeños, la cantidad de memoria que se puede necesitar puede dispararse sustancialmente.

Sin embargo, para la mayoría de los casos, existe un serio inconveniente. A la hora de afrontar la resolución de un problema de claves o generadores minimales, no es posible, en primera instancia, prever cuál va a ser la magnitud que va a alcanzar la resolución del problema (en términos de número de nodos y tiempo de ejecución) a la vista únicamente de la información de entrada. Esto obliga a realizar una serie de pruebas previas para configurar adecuadamente el entorno de ejecución en cuanto a número de procesadores, cantidad de memoria y espacio de almacenamiento (véase [10]). Eliminar la necesidad de dichas pruebas previas constituye sin duda un problema complejo de cara a futuras investigaciones, como se mencionará más adelante en el apartado de trabajos futuros.

Para evaluar las implementaciones realizadas (partiendo del hecho de que son implementaciones de algoritmos con el mismo orden de complejidad), no es suficiente con la comparación de los tiempos de ejecución de los métodos ya que este parámetro va a venir condicionado por la arquitectura *hardware* utilizada para llevar a cabo los experimentos. En este sentido, se han utilizado métricas adicionales: el número de nodos y el número de claves o generadores redundantes, que reflejan el tamaño del problema y la cantidad de cálculo superfluo realizado, respectivamente. Como muestran los experimentos presentados en [9–11, 13, 14], gracias a estas métricas se ha podido comprobar que, con el trabajo realizado en esta tesis, se ha conseguido en diferentes experimentos una reducción del número de nodos y/o del número de cálculos superfluos en un rango del 10-70 %.

En cuanto a los SRs conversacionales, el trabajo realizado se ha enfocado en el uso de implicaciones y la lógica para subsanar determinados problemas que aparecen en este tipo de SR. En concreto, se ha abordado el denominado problema de la dimensionalidad que surge cuando se trabaja con una cantidad muy elevada de atributos, lo que dificulta la interacción del sistema con el usuario.

En este campo también son varias las conclusiones alcanzadas. A alto nivel, la más importante es que, efectivamente, el tratamiento de la información realizado por medio de implicaciones y la lógica SL_{FD} puede aplicarse con éxito al campo de los SRs. Este hecho ya era auspiciado por

la existencia de trabajos en la literatura de SRs que utilizan conceptos de FCA [70,71,109,139]; el trabajo en esta tesis refuerza esta línea, en concreto para los SRs conversacionales, proponiendo nuevos métodos con los que abordar problemas comunes de esos sistemas y mejorar las aproximaciones existentes.

Más específicamente, se ha aportado una novedosa aplicación del algoritmo del cierre 2.2 para afrontar el problema de la sobrecarga de la información. Así, un punto importante de esta aportación ha sido la de aprovechar el resultado dual que resulta del algoritmo del cierre 2.2 , pues además del conjunto cerrado de atributos, el hecho de poder disponer del conjunto de implicaciones que recoge la semántica del sistema complementario a dicho conjunto, es aprovechado para volver a calcular de nuevo el conjunto de implicaciones, lo que tiene un coste exponencial.

La solución propuesta presenta un proceso conversacional de selección de elementos por parte del usuario a partir de los atributos de éstos. Este trabajo combina además características de sistemas basados en contenido con sistemas de recomendación basados en conocimiento mediante una gestión inteligente de las implicaciones teniendo como base el cierre 2.2.

Para concretar las conclusiones obtenidas, debe tenerse en cuenta que, tal y como se introdujo anteriormente en la Sección 1.3, existen numerosas opciones a la hora de evaluar el funcionamiento de un SR. Esto es razonable en tanto en cuanto el número de técnicas diferentes con las que trabajan los SRs es igualmente alto. Por tanto, es fundamental decidir qué métricas son oportunas de aplicar dependiendo del tipo de SR que se desee evaluar, pues evidentemente habrá casos en los que una métrica no tenga cabida para un tipo de SR determinado.

Dada la naturaleza del sistema desarrollado, las métricas que se han utilizado para evaluar el rendimiento están directamente relacionadas con el proceso de diálogo, como son el número de pasos de la conversación o el filtrado de atributos que se produce. Estas métricas devuelven resultados muy prometedores en ambos casos:

(i) En cuanto al número de pasos del diálogo, se puede ver como en gran

parte de los experimentos realizados, la conversación necesita menos de 3 ó 4 pasos para alcanzar recomendaciones adecuadas aun cuando el número total de atributos sea considerable (más de 30 atributos). Estos buenos resultados quedan contrastados en [15], donde se han realizado pruebas con los conjuntos de implicaciones derivados de *datasets* con 100 atributos diferentes y en las que la mayoría de las conversaciones finalizan en 2 ó 3 pasos.

(II) En cuanto al filtrado de atributos, los resultados son igualmente notables. Las pruebas mostradas en ese mismo artículo, demuestran cómo el uso del sistema conversacional ha evitado que el usuario tenga que interactuar a lo largo del diálogo con el 5-20 % de atributos (en el peor y mejor caso, respectivamente) reduciendo de esta forma la sobrecarga de información y mejorando la experiencia de usuario.

Estos resultados superan con creces los encontrados en la literatura con métricas de evaluación comparables, como [118] donde, incluso en pruebas con *datasets* con un menor número de atributos, se puede ver que por un lado, necesitan un número de pasos mayor, y por otro, que la reducción del número de atributos en la conversación es menor.

Para finalizar, se quiere destacar que las aplicaciones y métodos desarrollados para los tres campos son capaces de ir más allá del ámbito académico o de investigación. Las pruebas realizadas demuestran la viabilidad y la utilidad que las propuestas pueden aportar en entornos empresariales. Como muestras de ello, se pueden considerar como referencias los experimentos satisfactorios llevados a cabo sobre datos reales, como el caso de MovieLens y los repositorios de la UCI[1] para claves y generadores minimales, o el caso de la información real sobre enfermedades y fenotipos extraída de HPO[2] y OMIM[3] en el caso de los sistemas de recomendación conversacional.

[1] Universidad de California, Irvine (https://archive.ics.uci.edu/ml/index.php)
[2] Human Phenotype Ontology Consortium (https://hpo.jax.org/app/)
[3] Online Mendelian Inheritance in Man (https://https://www.omim.org)

6.2. Trabajos Futuros

Los resultados obtenidos motivan una serie de líneas de trabajo importantes para futuras investigaciones.

Respecto a las aportaciones referentes al tema de claves y generadores minimales existen varios aspectos con los que continuar a partir de este trabajo de investigación. Se irán introduciendo uno a uno sin perjuicio de que el orden de aparición denote una mayor o menos importancia.

En el momento actual no es posible prever cuál va a ser la magnitud que va a alcanzar la resolución del problema a la vista de la información de entrada, en términos de tiempo de cómputo y de recursos computacionales. Esto va a constituir en la mayoría de los casos un serio inconveniente ya que, para la realización de aplicaciones o pruebas, esta circunstancia no permite adelantar los requisitos de tiempo y recursos computacionales que serán necesarios. A pesar de ello, se ha observado que ciertas características, como el cardinal de la premisa o la conclusión en la implicación, suelen vaticinar patrones de comportamiento similares. En este sentido, el trabajo que se está llevando a cabo es investigar la motivación teórica de estos hechos empíricos, con la intención de poder identificar características que pronostiquen la complejidad que alcanzará una determinada ejecución del proceso.

Otro camino muy importante para continuar la investigación tanto para claves como para generadores minimales consiste en ahondar en la optimización del valor de corte o BOV. Recuérdese que el BOV es el valor a partir del cual la ejecución secuencial del código paralelo termina y se forman los diferentes subproblemas que serán resueltos en paralelo. El hecho de establecer un valor de corte adecuado es una tarea realmente compleja. En estos momentos se está estudiando la forma de expansión que tiene el tableaux con la intención de optimizar el valor de corte de manera que equilibre el trabajo realizado por cada procesador.

Otra línea de investigación sobre la que se está trabajando busca profundizar en el hecho de que aumentar el número de procesadores para la resolución de un problema no siempre redunda en una mejora del rendimien-

to. Hay casos en los que aumentar los recursos utilizados puede ser incluso contraproducente como se ha demostrado en las publicaciones que avalan esta tesis [13, 14]. En otras palabras, los tiempos de ejecución de los experimentos pueden incrementarse al aumentar los recursos *hardware*. Este efecto se debe, generalmente, a que el problema original se disemina de manera excesiva entre los procesadores disponibles provocando que el tiempo requerido por las comunicaciones para combinar los resultados parciales y así construir el resultado final contrarresta la ganancia en rendimiento que ofrece la capacidad de cómputo adicional. Para abordar este problema, se está trabajando en descubrir aquellas cotas de recursos a partir de las cuales el beneficio decrece. Para ello, se están investigando estrategias de optimización de recursos *hardware* en entornos de HPC [3, 126].

Otra tarea por la que continuar es realizar un nuevo diseño de los algoritmos paralelos que permita establecer comunicación entre las diferentes resoluciones paralelas de un mismo problema, de forma que se pueda mejorar la reducción de los cálculos redundantes. Para el caso concreto del cálculo de los generadores minimales, este objetivo es muy importante y la razón es la siguiente. Como se ha mencionado en el Capítulo 1, el método que mejores resultados obtiene en cuanto a número de nodos y tiempos de ejecución es GenMinGen, sin embargo, hay que recordar que, por el momento, es un método secuencial debido al requisito de establecer comunicación entre las diferentes soluciones parciales. Según el curso actual de la investigación, va a ser necesario considerar la aplicación del paralelismo a nivel de *software* y analizar cómo puede combinarse con la estrategia actual, centrada en el paralelismo *hardware*. Por tanto, se ha comenzado por investigar trabajos relacionados en la literatura sobre este tema [34].

En relación a los SRs conversacionales, aparecen varios aspectos interesantes a tener en cuenta en el futuro próximo.

En primer lugar, sería esencial averiguar qué características del *dataset* (tamaño, sobreespecialización, dispersión, sinónimos, etc.) son las más influyentes en el rendimiento del proceso de recomendación. En busca de este objetivo, en estos momentos se ha comenzado por investigar la literatura al respecto [19, 122].

Existen más aspectos de los SRs que sería recomendable investigar para mejorar el sistema conversacional desarrollado. Tal puede ser el caso de proporcionar explicaciones que justifiquen las recomendaciones que el usuario recibe. Este es un aspecto muy importante en un SR, ya que ayuda a mantener un mayor grado de confianza del usuario en los resultados generados por el sistema [111]. De hecho, la aceptación de un SR mejora cuando los usuarios comprenden las fortalezas y limitaciones del SR [92].

En este sentido, es relevante considerar que al tratar con implicaciones, nuestro SR garantiza que los resultados cumplen plenamente con lo que pide el usuario. Sin embargo, consideramos interesante la generación de explicaciones para justificar la reducción de atributos que se lleva a cabo a lo largo del proceso conversacional, al margen de los solicitados explícitamente por el usuario, por la acción del algoritmo del cierre.

Por todo ello, en este momento se está investigando a partir de la literatura los tres diferentes estilos de explicaciones habituales en los SRs actuales, que son:

(I) Los basados en usuario y sus preferencias, donde las explicaciones se centran en justificar las recomendaciones realizadas argumentando afinidad entre usuarios y/o preferencias [132].

(II) Los basados en objetos. En este caso, las explicaciones del SR se razonan teniendo en cuenta atributos añadidas de los objetos, como por ejemplo, su historial de valoraciones.

(III) Los basados en características. Las explicaciones son similares al caso anterior, pero esta vez teniendo en cuenta las características intrínsecas de los objetos [86].

En el futuro próximo, el trabajo se centrará en el estudio del tercer tipo de explicaciones ya que, por el momento, los dos primeros tipos no tienen cabida en el sistema desarrollado, ya que no se utilizan ni preferencias de usuario ni valoraciones.

Índice alfabético

Índice de figuras

Índice de tablas

Anexo

Apéndice A

Closed sets enumeration: a logical approach

Proceedings of the 17th International Conference
on Computational and Mathematical Methods
in Science and Engineering, CMMSE 2017
4–8 July, 2017.

Closed sets enumeration: a logical approach.

F. Benito-Picazo[1], P. Cordero[1], M. Enciso[1] and A. Mora[1]

[1] *Universidad de Málaga, Andalucía Tech, Málaga, Spain*

emails: fbenito@lcc.uma.es, pcordero@uma.es, enciso@lcc.uma.es, amora@ctima.uma.es

Abstract

Closed sets are the basis for the development of the concept lattice, a key issue in formal concept analysis. The enumeration of all the closed sets is a complex problem, having an exponential cost. In addition to the closed set, it is very useful for applications to add the information of all the minimal generators for each closed set. In this work we explain how to approach this problem from a complete set of implication by means of a sound and complete logic.

Key words: Formal concept analysis, closed sets, minimal generator, logic.

1 Introduction

Formal concept analysis (FCA) is a theoretical and practical framework to store information and manage them [GW99]. Data is stored in a table, representing a binary relation between a set of objects and attributes. The success of FCA relies on its solid theoretical framework and a wide set of methods and techniques to extract the knowledge from this data and manipulate it. One outstanding representation of the knowledge is the concept lattice, built over the closed sets, considering the subset relation as the order relation. Such representation depict a overall view of the information with a very strong formalism, opening the door to use the lattice theory as a metatheory to manage the information [BDVG17].

If-then rules have been introduced in several areas, dressed with different clothes. Thus, in relational databases [Cod71] they are named Functional Dependencies, in FCA they are named Implications and in Logic Programming (fuzzy logic) [BV06a] they are named if-then rules. All this notions captures a very intuitive idea: when the premise occurs, then the conclusion holds. Nevertheless, their semantics are very different and they further use are

 ISBN: 978-84-617-8694-7

also distinct. In this work we consider implications as elements to describe the information and we design a method to enumerate all closed sets and their minimal generators.

The proposed method is an evolution of [CEMO12], where the authors introduce a logic-based method based on $\mathbf{SL}_{FD}$, a sound and complete logic for implications. That method works by traversing the set of implications and applying a set of inference rules, following a tree paradigm in its execution. In that method, an exhaustive search was developed, producing the intended result but with an improvable performance. Here, we propose the design of several pruning strategies to improve such performance. These strategies are motivated by the idea of avoiding the opening of full branches in the tree or reducing the size of the information in their nodes.

The rest of the work is organized as follows: in the following section we present $\mathbf{SL}_{FD}$ and the axiomatic system which constitutes the basis of the MinGen algorithm. In Section 3 we present the algorithm to enumerate the closed sets and minimal generators and summarize the strategies to improve its execution in practice. The work ends with a brief conclusion.

2 Logic for implications

First, the main notions related formal concept analysis needed in this work are presented.

A **formal context** is a triple $\mathbf{K} := (G, M, I)$ where G is a set of objects, M is a set of attributes and $I \subseteq G \times M$ is a binary relation between G and M such that, for $o \in G$ and $a \in M$, $o\ I\ a$ means that the object o has the attribute a. Then, two mappings are defined $(\ )' \colon 2^G \to 2^M$ defined for all $A \subseteq G$ as $A' = \{m \in M \mid g\ I\ m \text{ for all } g \in A\}$, and $(\ )' \colon 2^M \to 2^G$ defined for all $B \subseteq M$ as $B' = \{g \in G \mid g\ I\ m \text{ for all } m \in B\}$. We use the same symbol since no confusion arises. This pair of mappings is a Galois connection.

The composition of the intent and the extent mappings, and vice versa, introduces two closure operators $(\ )'' \colon 2^G \to 2^G$ and $(\ )'' \colon 2^M \to 2^M$. The notion of closed set (as a fixpoint of a closure operator) is defined as follows:

Definition 1 *A **formal concept** is a pair (A, B) such that $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. Consequently, A and B are closed sets of objects and attributes, respectively called extent and intent.*

In this work we focus on the attributes closed sets. A key point in this work is the notion of the minimal generator (mingen) [GW99], which provides a minimal representation for each closed set, and is defined as follows:

Definition 2 *Let $\mathbf{K} = (G, M, I)$ be a formal context and $A \subseteq M$. The set of attributes A is said to be a minimal generator (**mingen**) if, for all set of attributes $X \subseteq A$ if $X'' = A''$ then $X = A$.*

 ISBN: 978-84-617-8694-7

Remark that the above definition allows to characterize each closed set by means of a minimal subset to provide a canonical representation of the closed sets. Moreover, we would like to remark that such representation is not unique, since a given closed sets can have several minimal generators.

The notion of minimal generator can also be defined from the point of view of implications. They are expressions $A \to B$ where A and B are attribute sets. A context satisfies the implication $A \to B$ if every object that has all the attributes from A also has all the attributes from B.

Definition 3 *An (attribute) implication of a formal context* $\mathbf{K} = (G, M, I)$ *is defined as a pair* (A, B)*, written* $A \to B$*, where* $A, B \subseteq M$ *and* $A \cap B = \varnothing$*. Implication* $A \to B$ *holds (is valid) in* $\mathbf{K}$ *if* $A' \subseteq B'$*.*

The set of all valid implications in a context satisfies the well-known Armstrong's axioms [Arm74], which constitutes the pioneer logic to manage implications. The author introduces a sound a complete axiomatic system to infer new implications holding in a context from a given set of implications. Moreover, this logic constitutes a proposal to solve the attribute closure, i.e. to find the maximal set of attributes A^+ such that the implication $A \to A^+$ holds. As we mentioned, this maximal set is a closed set as defined before and this closure operator $()^+$ allows us to guide the automatization the search for closed sets. Thus, we introduce a new logic suitable for this goal.

The introduction of the Simplification Logic [MECF12], named $\mathbf{SL}_{FD}$, opened the door to the development of automated reasoning methods directly based on its novel axiomatic system. $\mathbf{SL}_{FD}$ considers reflexivity as axiom scheme

$$\texttt{[Ref]} \quad \frac{}{A \to A}$$

together with the following inference rules called Fragmentation, Composition and Simplification respectively.

$$\texttt{[Frag]} \ \frac{A \to BC}{A \to B} \qquad \texttt{[Comp]} \ \frac{A \to B,\ C \to D}{AC \to BD} \qquad \texttt{[Simp]} \ \frac{A \to B,\ C \to D}{A(C \setminus B) \to D}$$

Similarly to the dual vision of closed set (in terms of Galois connection and implications), a dual definition of minimal generator can be done in terms of implications:

Definition 4 *Let* $\mathbf{K} = (G, M, I)$ *be a formal context and* $A \subseteq M$*. The set of attributes* A *is said to be a minimal generator (**mingen**) if, for all set of attributes* $X \subseteq A$ *if* $X \to A^+$ *then* $X = A$*.*

In the following section we introduce the algorithm to enumerate the minimal generator based on $\mathbf{SL}_{FD}$ and describe the strategies to improve its performance.

 ISBN: 978-84-617-8694-7

3 An algorithm to enumerate all closed sets and their minimal generators

Simplification logic has allowed us to design several executable methods to manage implications. Thus in [MECF12] we developed a novel method to compute attribute closure strongly based on $\mathbf{SL}_{FD}$ inference rules. This method has been showed to have a better performance than the classical methods based on indirect techniques. One outstanding characteristics of $\mathbf{SL}_{FD}$ closure is the output it renders: given a set of attributes $X \subseteq M$ and a set of implications Γ, it renders its closure X^+ and a new set of implications Γ' which describes the remaining knowledge in the set $M \setminus X^+$.

This logic-based closure method is the basis of another method, named MinGen, to compute the set of all minimal generators from a set of implicant set presented in [CEMO12]. The algorithm works by applying the $\mathbf{SL}_{FD}$Closure algorithm to each implication in the set, opening a new branch. This application provides a new candidate to be added to mingen and a smaller implications set which guides us in the search of new sets of attributes to be added to mingens, producing a tree-like execution.

In summary, the input of this algorithm is a set of attributes M and a set of implications Γ over the attributes in M. The output is the set of closed sets endowed with all the minimal generators, i.e. $\{\langle C, mg(C)\rangle : C$ is a closed set of attributes$\}$ where $mg(C) = \{D : D$ is a mingen and $D^+ = C\}$. In this work we only consider non-trivial minimal generators, i.e. pairs of closed set and minimal generator $\langle X, Y\rangle$ where $Y \subsetneq X$.

For example, if $M = \{a, b, c, d, e, f\}$ and $\Gamma = \{a{\to}b, bc{\to}d, de{\to}f, ace{\to}f\}$ the output is the set $\{\langle abcdef, \{ace\}\rangle, \langle abdef, \{ade\}\rangle, \langle abcde, \{ac\}\rangle, \langle bcdef, \{bce\}\rangle, \langle bcd, \{bc\}\rangle, \langle def, \{de\}\rangle, \langle ab, \{a\}\rangle, \langle c, \{c\}\rangle, \langle \varnothing, \{\varnothing\}\rangle\}$. The execution of the method is depicted in Figure 1. We refer the reader to [CEMO12] for a detailed description of the method and its theoretical results.

In this work, we propose two pruning strategies to improve MinGen method. We briefly describe them as follows:

- The first strategy characterizes the branches that can be considered a superfluous one because all their nodes explore closed sets and minimal generators already considered in another branches. To implement this strategy we will consider a subset test on the branches of the same level and in the same branch.

- The second strategy is to expedite the execution of the method by including the closure method in each node in two steps, so that in the first one the closure set is computed and, in the second one, the resulting set of implications is computed taking into account this closed set. In this way, the resulting set of implications will be a smaller one and the method will have a better performance.

 ISBN: 978-84-617-8694-7

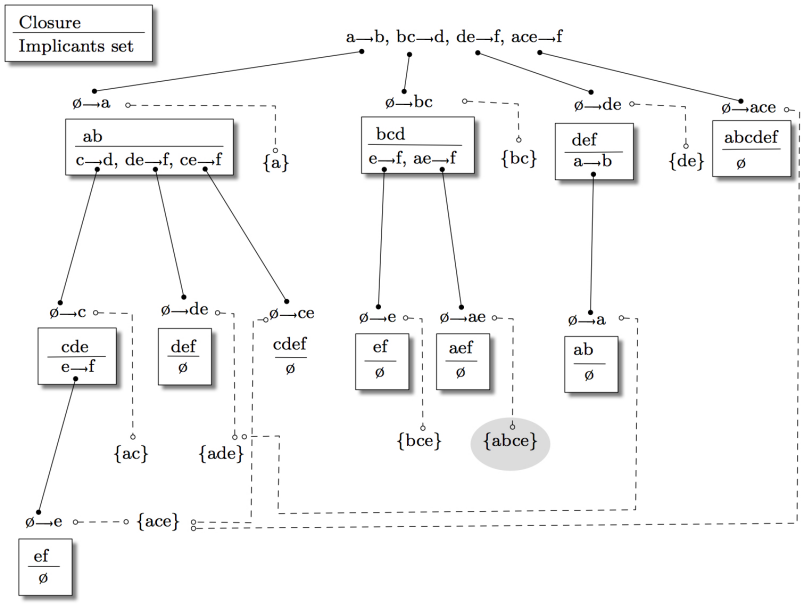

Figure 1: Exemplification of

 ISBN: 978-84-617-8694-7

4 Conclusion and future works

In this work we have studied the state of the art in the enumeration of closed sets and minimal generators based on logic. We have considered the MinGen method based on Simplification Logic as the target of our work and we propose to improve it by means of several prunes to improve its performance.

In a future work, we propose to establish the theoretical results to state these strategies and to develop an exhaustive practical experiment to show its benefits.

Acknowledgements

This work is partially supported by project TIN2014-59471-P of the Science and Innovation Ministry of Spain, co-funded by the European Regional Development Fund (ERDF).

References

[Arm74] W. Armstrong. Dependency structures of data base relationships. *Proceedings of the International Federation for Information Processing Congress*, pages 580–583, 1974.

[BV06a] R. Belohlávek and V. Vychodil. Computing non-redundant bases of if-then rules from data tables with graded attributes. In *2006 IEEE International Conference on Granular Computing*, pages 205–210, 2006.

[BDVG17] K. Bertet, C. Demko, J.-F. Viaud, and C. Guérin. Lattices, closures systems and implication bases: A survey of structural aspects and algorithms. *Theoretical Computer Science*, `http://dx.doi.org/10.1016/j.tcs.2016.11.021`, 2017.

[Cod71] E. F. Codd. Further normalization of the data base relational model. *IBM Research Report, San Jose, California*, RJ909, 1971.

[MECF12] A. Mora, M. Enciso, P. Cordero, and I. Fortes. Closure via functional dependence simplification. *International Journal of Computer Mathematics*, 89(4):510–526, 2012.

[CEMO12] P. Cordero, M. Enciso, A. Mora, and M. Ojeda-Aciego. Computing minimal generators from implications: a logic-guided approach. In *Proceedings of the Ninth International Conference on Concept Lattices and Their Applications*, pages 187–198, 2012.

[GW99] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag New York, Inc., 1999. Translator-C. Franzke.

 ISBN: 978-84-617-8694-7

Apéndice B

Conversational recommendation to avoid the cold-start problem

Proceedings of the 16th International Conference
on Computational and Mathematical Methods
in Science and Engineering, CMMSE 2016
4–8 July, 2016.

Conversational recommendation to avoid the cold-start problem

F. Benito-Picazo[1], M. Enciso[1], C. Rossi[1] and A. Guevara[1]

[1] *Department of Languages and Computer Science, Universidad de Málaga, Andalucía Tech, Málaga, Spain*

emails: fbenito@lcc.uma.es, enciso@lcc.uma.es, rossi@uma.es, guevara@uma.es

Abstract

Recommender systems has become a widespread topic, allowing to connect user demands to those products more suitable to their preferences. The more information we provide to the system, the better the system works. This is a weak point of recommenders: they need an initial information belonging to each new user. In this paper we propose to avoid the so-called cold-start problem by using a conversational recommendation approach. We consider products characteristics as attributes and deal with the attribute implications by means of the simplification logic to guide the user in the search.

Key words: Recommendation systems, conversational recommendation, logic, implications

1 Introduction

Nowadays recommender systems have established a solid field of knowledge within information technologies. They are a kind of software that group together a wide range of techniques and applications with the aim of providing the best user experience [18]. There has been much progress done towards recommender systems during last decade [1] but there is still so much work remained. Examples of the applications concerning recommender systems go over many different topics of today's society such as recommending books, music, documents, e-commerce, tourism, medical diagnosis, among others. Recommender Systems constitute a hot topic indeed, as we can notice by the way in which many top companies worldwide spend their efforts and resources developing more and better systems for them to

increase their benefits. By these means, companies with absolutely different market niches delegate their most important duties to recommender systems due to the wide range of possibilities they offer; and yet companies selling products are not all of them, global leaders in other fields as totally come aboard with recommender systems by recommending new friends, groups, followers, and other social connections.

When recommendations are based on of the element evaluation made by other users or by similarity between the user preferences and these characteristics, recommender systems need to face many problems before they can flow into good recommendations. The first one we need to remark is the well-known cold-start problem [10], that appears when recommender systems try to elaborate reliable recommendations from the absence of initial information. Cold-start problem may be handled by requesting other agents to share what they have already learned from their respective users [11]. Also, new items (those which have not received any ratings from the community yet) would be assigned a rating automatically, based on those given by the community to other similar items [20] and so, we are at the mercy of similarity rules. In the same direction, until the new element has not been evaluated by a significant number of users, the system will not be able to recommend it. An item that is not recommended remain unnoticed by most of the user community, thus, we can enter into a vicious circle in which a set of elements of the recommender systems will be left out of the rating process and/or recommendations continuously [16]. In most of cases, users do not rate all the features we would desire for the optimum running of the recommender systems, this reveals scarcity problem.

In this work, we propose to deal with the cold-star problem by introducing an information flow based on the dialogue with the user. The lack of initial information is avoided with the design of a process with allows to collect this information of the user and storing them for further access to the system. This process, as we shall see, is a recommender-like system, to allow the user for getting some usefullness in its use.

2 Recommender systems and the conversational issue

There exists different kinds of recommender systems usually classified on how recommendations are made [1]. The most known and extended ones are collaborative filtering, content-based and demographic systems. Besides, in recent years there has been a great expansion of context-aware recommender systems [2] and knowledge-based recommender systems [14]. Other group of recommender systems that worths to be considered is that one focused on recommendations involving group of people [9]. Collaborative filtering systems [13], recommend items that other users have already rated before. Recommendations made by content-based systems present items similar to the ones the user preferred in the past [12]. Context-aware recommender systems try to adapt their recommendations to the world around the user. Finally, knowledge-based approaches are different; they manage functional

knowledge about how an item matches a particular need, and they can therefore reason about the relationship between a need and a possible recommendation. These characteristics make knowledge-base recommender systems not only valuable systems on their own, but also highly complementary to other types of recommender systems. However, the history of recommender systems has broadly demonstrated that best strategies are those who merge characteristics from different kinds of recommender systems in order to generate hybrids conforming best features of each one [6, 4].

In general, most of widely used recommendation techniques requires information to build a user profile before generating a result. In some cases, that information may be gathered explicitly: for example, requiring data about age, gender, etc. during a registration process, or by means of ratings and opinions about the recommended items. In other cases, the system may get implicit information from the browsing and/or purchase user history.

Nevertheless, there are contexts in which this previous information it is not available. This is the case of the well-known cold-start problem, when a new user asks for his first recommendation and obviously the system has not any information about him. This situation also occurs in systems where users make occasional use.

An interesting approach to solve this problem us the use of the so-called conversational recommender systems [7, 8]. These are closely related with critiquing recommender systems [17, 21]. In these works, recommendation is enriched by means of a dialog with the user that allows an incremental elicitation of his preferred item features. To promote an effective use of this approach, our proposal produces as an output a recommendation only based on the user dialogue information. In this way, the system is attractive for those user that are new in the system and can be used as a preliminary system to store user preferences for further accesses.

3 A logic approach to conversational recommendation

Our proposal to integrate recommender systems and the conversational issue is based on a sound and complete logic. As we shall see, such an strong basis allows us to include a reasoning method in the process and allows us to store the information in a natural way to be managed in the future by knowledge-base recommenders.

We built our framework on a basic elements, the implications. They correspond to formulas $a_1 \wedge \ldots \wedge a_n \rightarrow b_1 \wedge \ldots \wedge b_m$. The propositions $a_1, \ldots a_n, b_1, \ldots, b_m$ are elements of a set Ω and they are interpreted as properties concerning attributes. For this reason, propositional symbols are named attributes. To compact notation it is usual to denote the above formulas as $A \rightarrow B$ being $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_m\}$ i.e. sets of attributes are conjunctively interpreted.

The symbolic management of implications was originally proposed in [3]. However, due to the central role that transitivity plays in this axiomatic system, the development

of executable method to solve implications problems has rest on indirect methods. For instance, the proposal to solve the attribute closure, i.e. to find the maximal set of attributes A^+ such that the implication $A \to A^+$ holds has been traditionally tackle by using a basic method which exhaustively uses the subset relation to add new elements in the conclusion.

The introduction of the Simplification Logic, named $\mathbf{SL}_{FD}$, [5] opened the door to the development of automated reasoning methods directly based on its novel axiomatic system. $\mathbf{SL}_{FD}$ considers reflexivity as axiom scheme

$$\texttt{[Ref]} \quad \frac{}{A \to A}$$

together with the following inference rules called Fragmentation, Composition and Simplification respectively.

$$\texttt{[Frag]} \ \frac{A \to BC}{A \to B} \qquad \texttt{[Comp]} \ \frac{A \to B,\ C \to D}{AC \to BD} \qquad \texttt{[Simp]} \ \frac{A \to B,\ C \to D}{A(C \setminus B) \to D}$$

Later, in [15] we presented an attribute closure method closely tied to the Simplification logic axiomatic system. Apart from having a strong base, the main advantage of our method is that its output is twofold: besides the maximal set constituting the closure of the input, it also renders a reduced set of implications which enclose the semantics that is outside the set A^+. We would like to remark that this two inputs are computed in linear time, overtaking the hard cost of a data mining process if we were interested in extracting the new set of implication for the reduced dataset after each search step.

This characteristics provides a key information to further inferences in an iterative search process. This is the core of our proposal to design a conversational recommendation based on our attribute closure operator. The recommendation process will go along the following points:

0. We depart from the premise that we have a dataset containing items and attributes, and the set of implications that holds on it. This is considered point zero and, as we have mentioned, it does not requieres any information from the user to be started.

1. Once we count on this information, the user interacts with the system by selecting an attribute we wish an item to fit.

2. Then, the process flows into the closure algorithm calculating both the set closure for this attribute and above all, the set of implications that remains outside the closure and complete them.

3. Once the closure algorithm has finished, a new reduced dataset is shown. At this point, we can stop the interaction whether we are already satisfied with the result or we can go ahead trying to get a more suitable recommendation. The improvement here goes as follows. For further queries, we have reduced the number of available attributes

 ISBN: 978-84-608-6082-2

deleting those included in the closure set. Even that this could be accomplished by classic closure algorithms, the major point of our method is that, *at the same time*, we also reduce the number of implications, and so, in every refining-attempt we do not need to start the process form the beginning but continuing from here, where both attributes and implications have been decreased. Consequently, the process maintains its linear complexity and the interaction becomes truly faster.

4. In this way, we select a new attribute and resume the search.

5. We carry on selecting attributes until we get a satisfying recommendation or we run out of attributes.

4 Conclusion and future works

In this paper we propose to approach the cold-start problem. We mining the dataset containing the product information to get a set of attribute implications. This set is managed by using the inference system of simplification logic to guide the search of new users.

As a future work, we propose to study the impact of simplification closure in the performance of our approach. Our method allows to get, in an iterative way, intermediate closure set of attributes and the corresponding reduced set of implications. This characteristics allows to proceed step by step and, at the same time, accelerate the search.

Acknowledgements

This work is partially supported by project TIN2014-59471-P of the Science and Innovation Ministry of Spain, co-funded by the European Regional Development Fund (ERDF).

References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.

[2] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In Ricci et al. [19], pages 217–253.

[3] W. W. Armstrong. Dependency structures of data base relationships. In *IFIP Congress*, pages 580–583, 1974.

[4] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutirrez. Recommender systems survey. *Knowledge-Based Systems*, 46(0):109 – 132, 2013.

[5] P. Cordero, M. Enciso, A. Mora, and I. P. de Guzmán. $\mathrm{Sl}_{\mathrm{fd}}$ logic: Elimination of data redundancy in knowledge representation. *IBERAMIA*, pages 141–150, 2002.

[6] L. M. de Campos, J. M. Fernndez-Luna, J. F. Huete, and M. A. Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *International Journal of Approximate Reasoning*, 51(7):785 – 799, 2010.

[7] S. Guerrero and M. Salamo. Increasing retrieval quality in conversational recommenders. *IEEE Transactions on Knowledge and Data Engineering*, 24(10):1876–1888, 2012.

[8] D. Jannach and G. Kreutler. Rapid development of knowledge-based conversational recommender applications with advisor suite. *Journal of Web Engineering*, 6(2):165–192, 2007.

[9] H.-N. Kim and A. El-Saddik. A stochastic approach to group recommendations in social media systems. *Information Systems*, 50(0):76 – 93, 2015.

[10] H.-N. Kim, A. El-Saddik, and G. Jo. Collaborative error-reflected models for cold-start recommender systems. *Decision Support Systems*, 51(3):519–531, 2011.

[11] Y. Lashkari, M. Metral, and P. Maes. Collaborative interface agents. In *In Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 444–449. AAAI Press, 1994.

[12] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In Ricci et al. [19], pages 73–105.

[13] M. Maleszka, B. Mianowska, and N. T. Nguyen. A method for collaborative recommendation using knowledge integration tools and hierarchical structure of user profiles. *Knowl.-Based Syst.*, 47:1–13, 2013.

[14] M. Mandl, A. Felfernig, E. Teppan, and M. Schubert. Consumer decision making in knowledge-based recommendation. *Journal of Intelligent Information Systems*, 37:1–22, 2011.

[15] A. Mora, P. Cordero, M. Enciso, I. Fortes, and G. Aguilera. Closure via functional dependence simplification. *Int. J. Comput. Math.*, 89(4):510–526, 2012.

[16] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 21–28, New York, NY, USA, 2009. ACM.

[17] J. Reilly, K. McCarthy, L. McGinty, and B. Smyth. Incremental critiquing. *Knowledge.-Based Systems*, 18(4-5):143–151, 2005.

[18] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[19] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[20] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 253–260, New York, NY, USA, 2002. ACM.

[21] W. Trabelsi, N. Wilson, D. Bridge, and F. Ricci. Preference dominance reasoning for conversational recommender systems: a comparison between a comparative preferences and a sum of weights approach. *International Journal on Artificial Intelligence Tools*, 20(4):591–616, 2011.

Apéndice C

Keys for the fusion of heterogeneous information

Proceedings of the 15th International Conference
on Computational and Mathematical Methods
in Science and Engineering, CMMSE 2015

Keys for the fusion of heterogeneous information

Benito-Picazo, F.[1], Cordero, P.[1], Enciso, M.[1] and Mora, A.[1]

[1] *Universidad de Málaga, Andalucia Tech, Spain*

emails: fbenito@lcc.uma.es, pcordero@uma.es, enciso@lcc.uma.es, amora@ctima.uma.es

Abstract

The management of heterogeneous information is a current topic which demands the use of intelligent techniques to deal with data semantics. In this work we approach this problem by using Simplification Logic. It has a sound and complete inference system conceived to treat implications and functional dependencies. The automatic processing of functional dependencies allows to develop methods and tools to tackle most classical problems in database and information processing. In this work, we use Simplification Logic to design a method to enumerate all minimal keys of a data repository inferring them from a set of functional dependencies. We also illustrates how this method provides a successful way to solve some outstanding problems in data processing in linked data.

Key words: Heterogenous information, integration, keys, logic.

1 Introduction

The notion of a key is fundamental to any data model, including Codd's relational model of data [Codd, 1970]. A key of a relation schema is composed by a subset of attributes playing the role of a *domain* in a given function whose *image* is the whole set of attributes. This way, the table is viewed as its extensional definition. These functions are described by means of a *Functional Dependency (FD)* which specifies a constraint between two subset of attributes, denoted $A \to B$, ensuring us that for any two tuples in a table, if they agree on A, they also agree on B.

Besides primary keys, unique constraints (candidate keys) provide a more complete understanding of the model. Keys are not only fundamental to data design, they are also considered powerful tools to solve several problems related to a lot of fields of information management as mentioned before. Indeed, in [Sismanis et al., 2006] the authors affirm

that "*identification of keys is a crucially important task in many areas of modern data management, including data modeling, query optimization (provide a query optimizer with new access paths that can lead to substantial speedups in query processing), indexing (allow the database administrator to improve the efficiency of data access via physical design techniques such as data partitioning or the creation of indexes and materialized views), anomaly detection, and data integration*".

Identifying properly the keys of a relation schema is a crucial task for a lot of modern areas of information management (data modeling [Simsion and Witt, 2005], query optimization [Kemper and Moerkotte, 1991], indexing [Manolopoulos et al., 1999], etc).

A very outstanding characteristic of keys is their minimality. We denote a key as minimal when every attribute contained in its attribute set is necessary to keep the property of key, i.e. keys with no superfluous attributes. Thus, in the literature, the key finding problem is focused on minimal keys. Giving an attribute set A, the cardinality of the set 2^A forces us to consider in applying special techniques that lead the search of the sets being candidates to become minimal keys.

Keys constraints specify the sets of attributes of a relation such that their projection univocally identifies each tuple of the relation. The key finding problem consists in finding all the attribute subsets which make up a minimal key as of a set of FDs occurring within a schema of a relational model table.

In Table 1, we illustrate its semantics by the following basic example.

Table 1: Movie table

Title	Year	Country	Director	Nationality	Star
Pulp Fiction	1994	USA	Quentin Tarantino	USA	John Travolta
Pulp Fiction	1994	USA	Quentin Tarantino	USA	Uma Thurman
Pulp Fiction	1994	USA	Quentin Tarantino	USA	Samuel L. Jackson
King Kong	2005	New Zealand	Peter Jackson	New Zealand	Naomi Watts
King Kong	2005	New Zealand	Peter Jackson	New Zealand	Jack Black
King Kong	1976	USA	De Laurentiis	IT	Jessica Lange
King Kong	1976	USA	De Laurentiis	IT	Jeff Bridges
Django Unchained	2012	USA	Quentin Tarantino	USA	Jamie Foxx
Django Unchained	2012	USA	Quentin Tarantino	USA	Samuel L. Jackson

From the information in Table 1, we may ensure that the following FDs are satisfied: $Title, Year \rightarrow Country$; $Title, Year \rightarrow Director$; $Director \rightarrow Nationality$. Moreover, the table has only one minimal key: $\{Title, Year, Star\}$

It is imperative to state that though we are dealing with FDs, it is not the responsibility of this work to extract them from a relation schema. There are several techniques that provide us this work [Huhtala et al., 1999], [Yao et al., 2002]. Therefore, we will begin with given sets of FDs and then go ahead finding minimal keys within them.

In the same way, it is necessary to clarify that what we are researching is not a matter of data mining techniques [Fayyad et al., 1996]. Giving an overall view, data mining could be considered as a computational process of discovering patterns in large data sets and its

 ISBN: 978-84-617-2230-3

goal goes to extract information from a data set and transform it into an understandable structure for further use [Witten et al., 2011]. Nevertheless, what we are studying and developing are mechanisms and algorithms for deriving all minimal keys so they can help us performing a more intelligent and efficient way to manage the information stored in relational schemes.

In this work we will concentrate our efforts on those algorithms guided by logic, and most specifically, those using Tableaux paradigm [Morgan, 1992] [Risch and Schwind, 1992] for deriving keys of a relation schema using inference systems.

The problem of the key finding methods arises when we try to deal with a huge amount of information since the tableaux's building mechanism produces such an explosion of the search space that we go beyond the machine capabilities, even with small problems.

2 Background

Before going further, we need to introduce the necessary terms from relational database theory. Due to space limitation, we refer those readers non familiar with the basic notions of FDs and relational databases to previously visit [Elmasri and Navathe, 2010]. The notion of FDs are well-known in other areas as Formal Concept Analysis with the concept of implications.

Definition 1 (Attribute) *An attribute a is an identifier for an element of some domain D. We use letters $a, b, c, d, ...$ for attributes. Let U be a set of attributes. An attribute set X over U is a subset of U. We use capital letters X, Y, Z, V, ... for attribute sets.*

Definition 2 (Functional dependency) *Let U be a set of attributes. A functional dependency (FD) over U is an expression of the form $X \to Y$, where X, Y are attribute sets. It is satisfied in a table R if for every two tuples of R, if they agree on X, then they agree on Y.*

Definition 3 (Relation schema) *A relation schema $R = \langle U, F \rangle$ is an ordered pair consisting of an attribute set U and a set F of FDs over U.*

A key of a relational table is a subset of attributes that allows us to uniquely characterize each row. It may be defined by means of FDs as follows:

Definition 4 (Key) *Given a table R over the set of attributes U, we say that K is a key in R if the functional dependency $K \to U$ holds in R.*

Definition 5 (Minimal Key) *Given the table R, the attribute set $K \subset U$ is said to be a minimal key if it is a key of R and for all attribute $k \in K$ the subset $K - \{k\}$ is not a key of R.*

 ISBN: 978-84-617-2230-3

3 Simplification Logic for finding-key problem

3.1 SL_{FD}

Now, we summarize the axiomatic system of SL_{FD} [Cordero et al., 2013]. The inference system for SL_{FD} is equivalent to the well-known Armstrong's axioms as the first complete inference system for functional dependencies. It avoids the use of transitivity and is guided by the idea of simplifying the set of FDs by removing redundant attributes efficiently.

SL_{FD} is defined as the pair (L_{FD}, S_{FD}) where S_{FD} has the following axiom scheme and inference rules. The third rule is named Simplification rule and it is the core of SL_{FD}:

$$[Axiom] \quad \frac{Y \subseteq X}{X \to Y}$$

$$[Frag] \quad \frac{X \to Y}{X \to Y'}, \quad [Comp] \quad \frac{X \to Y \quad U \to V}{XU \to YV}$$

$$[Simp] \quad \frac{X \to Y \quad U \to V}{(U - Y) \to (V - Y)'} \quad X \subseteq U, X \cap Y \neq \emptyset$$

3.2 SST Method

The method we will use to find all minimal keys in schemes provided from two heterogeneous sources is SST method (see [Cordero et al., 2014] for more details). The input of the tableaux method is a set of attributes Ω and a set of formulas F. We build a tree as follows:

1. The root of the tree will be $(\Omega \to \Omega, F)$.
2. For each node $(U \to \Omega, F)$ with $U \neq \emptyset$ and each minimal formula $A \to B \in F$, a new children node $(U' \to \Omega, F')$ is added where:
   - $U' \to \Omega$ is obtained applying [lSimp] to $A \to B$ and $U \to \Omega$. That is, $U' = A \cup (U \setminus B)$.
   - F' is computed by applying [sSimp] to $A \to B$ and every formula in F. Moreover, in the new set of formulas, Union equivalence is applied and degenerated formulas are removed.
3. The method renders $Minimal\{U | (U \to \Omega, \emptyset)$ is a leaf of the tree$\}$.

 ISBN: 978-84-617-2230-3

4 Applications of keys in Linked Data

There are a lot of fields of knowledge in computer science where counting on efficient techniques for data management is crucial. Generally, it is an engineering work to establish and choose the keys as a part of the normalization process of the schema. The challenge is to figure out those attributes of the schema that allow identifying univocally each tuple of the relation. Lets show an easy example.

Example 1 *Assume that we have stored in a table the main data of an enterprise crew such as: Name, Age, ID Number, Phone Number. A priori, we notice several alternatives in order to identify each employee.*

As a key we could consider: ID Number, the pair (Name, Phone Number) or even (ID Number, Name). However, from all of that possibilities, ID Number arises as the best one since the other ones contain information that is not absolutely necessary to identify each person. Two people with different names could share the same phone number (members of a same home will share the same phone number). Pairs (ID Number, Name) will also identify properly each person, but the Name attribute is not indispensable since ID Number will be definitely enough as identification.

So, even in this trivial example we can easily notice that deriving keys is a task with an indisputable importance. Lets go deeper in details with a bit more realistic problems.

Linked Data is about connecting pieces of related data and information coming from different sources. In computing, linked data describes a method of publishing structured data so that it can be interlinked and become more useful. It builds upon standard Web technologies such as HTTP, RDF [1] and URIs, but rather than using them to serve web pages for human readers, it extends them to share information in a way that can be read automatically by computers. This enables data from different sources to be connected and queried [Bizer et al., 2009].

A typical case of a large Linked Dataset is DBPedia, which, essentially, makes the content of Wikipedia available in RDF. The importance of DBPedia is not only that it includes Wikipedia data, but also that it incorporates links to other datasets on the Web, e.g., to Geonames. By providing those extra links (in terms of RDF triples) applications may exploit the extra (and possibly more precise) knowledge from other datasets when developing an application; by virtue of integrating facts from several datasets, the application may provide a much better user experience.

For instance, the DBpedia resource for Brussels (http://dbpedia.org/resource/Brussels) can be linked to the one maintained by the Statistics Belgium

[1] RDF, the Resource Description Framework, is one of the key ingredients of Linked Data, and provides a generic graph-based data model for describing things, including their relationships with other things. RDF data can be written down in a number of different ways, known as serialisations. Examples of RDF serialisations include RDF/XML, Notation-3 (N3), Turtle, N-Triples, RDFa, and RDF/JSON.

(http://location.testproject.eu/so/au/AdministrativeUnit/STATBEL/24000).

Linking these two data resources allows us to get richer information about Brussels. Besides that, the attributes of the data entity for the city of Brussels is country. This attribute reveals that a city is positioned in/belongs to a country. In our case the value for country is Belgium. There are different options for encoding this information.

One way would be to include the value for country as text, e.g. a literal or a string. This option however cannot take us too far and can suffer from different writings, different languages and even spelling errors. The Linked Data approach in this case opts for replacing the text value with a URI pointing to the specific country, i.e. to Belgium (the URI of DBpedias resource for Belgium is http://dbpedia.org/resource/Belgium). The Linked Data option allows us to unambiguously refer to Belgium and also navigate through the links in order to collect more information about Brussels.

Due to these considerations, we need to find out the minimal set of properties necessary to make up the appropriate connections between the data in both schemes. Therefore, we need the minimal keys from each one of the sets so we can decide later which element will be our joining one.

In the following, we outline how the heterogeneous information is linked in two dataset with information about films.

Example 2 *The repositories* `http://www.imdb.com` *and* `https://www.filmaffinity.com` *contain information about films with different structure. The goal is to obtain the keys in order to connect the knowledge stored in both datasets.*

The method to make the fusion of information is summarized in the next items:

- *To select a group of films.*
- *To take the topics from* IMDB repository.
- *To extract the implications using the tool for Formal Concept Analysis named* ConceptExplorer.
- *To calculate the minimal keys for IMBD schema.*
- *To take the topics from* Filmafinity repository.
- *To extract the implications using the tool for Formal Concept Analysis named* ConceptExplorer.
- *To calculate the minimal keys for Filmafinity schema.*
- *To repeat the same with the joining of both implication sets.*

The result of applying SST Method to this two heterogeneous dataset is the following:

 ISBN: 978-84-617-2230-3

```
 IMDB
Topics: Action Comedy Crime Drama Fantasy Romance Thriller

Little City (1998) Comedy Romance
Driver, The (1978)  Action Crime Thriller
Father of the Bride (1950) Comedy Romance
Bio-Dome (1996) Comedy
Fast Runner, The (2001) Drama Fantasy
Overboard (1987) Comedy Romance
Get Rich or Die Tryin? (2005) Crime Drama

Implications:
Romance -> Comedy
Thriller -> Action Crime
Action -> Crime Thriller
Keys:
Romance Thriller
Romance Action

Filmaffinity
Topics: Action Comedy Crime Drama Family Film-noir Nature Romance Survival

Little City (1998) Comedy Drama
Driver, The (1978)  Action Crime Film-noir
Father of the Bride (1950) Comedy Romance Family
Bio-Dome (1996) Comedy
Fast Runner, The (2001) Drama Nature Survival
Overboard (1987) Comedy
Get Rich or Die Tryin? (2005) Drama

Implications:
Action -> Crime Filmnoir
Crime -> Action Filmnoir
Family -> Comedy Romance
Filmnoir -> Action Crime
Romance -> Comedy Family
Survival -> Drama Nature
Nature -> Drama Survival
```

 ISBN: 978-84-617-2230-3

```
Keys:
Action Family Survival
Action Family Nature
Action Romance Survival
Action Romance Nature
Family Filmnoir Survival
Family Filmnoir Nature
Filmnoir Romance Survival
Filmnoir Romance Nature
Crime Family Survival
Crime Family Nature
Crime Romance Survival
Crime Romance Nature

The joining of both topics:
Topics: Action Comedy Crime Drama Family Fantasy Film-noir Nature Romance Survival Thrill

Little City (1998) Comedy Drama Romnace
Driver, The (1978)  Action Crime Film-noir Thriller
Father of the Bride (1950) Comedy Romance Family
Bio-Dome (1996) Comedy
Fast Runner, The (2001) Drama Fantasy Nature Survival
Overboard (1987) Comedy Romance
Get Rich or Die Tryin? (2005) Crime Drama

Implications:
Comedy Drama ->  Romance
Family -> Comedy Romance
Fantasy ->  Drama Nature Survival
Nature ->  Drama Fantasy Survival
Romance ->  Comedy
Survival ->  Drama Fantasy Nature
Thriller ->  Action Crime Filmnoir
Action ->  Crime Filmnoir Thriller
Filmnoir ->  Action Crime Thriller

Keys:
Family Filmnoir Fantasy
Family Action Fantasy
```

 ISBN: 978-84-617-2230-3

```
Family Thriller Fantasy
Family Filmnoir Nature
Family Action Nature
Family Thriller Nature
Family Filmnoir Survival
Family Action Fantasy
Family Thriller Survival
```

5 Conclusions

The fusion of heterogeneous information is an emergent problem in which the use of Logic is adequate in order to incorporate automated reasoning mechanism. We have proposed the use of Simplification Logic to manipulate functional dependencies (implications).

SST Method is based of Simplification Logic and allow us to enumerate all minimal keys of a data repository inferring them from a set of functional dependencies. We illustrate how the method can be used to solve problems related with the integration/fusion of heterogeneous information in linked data.

For a extension of this work, we are envolved in the introduction of parallelism techniques in the implementation of the algorithms that could help us dealing with problems containing a substantial amount of information at the input. We think that the tableaux paradigm used in logic matches perfectly the design of the parallel versions of the algorithms. This strategy could be used in order to resolve complex input problems. We truly need to go further searching for strategies in order to find out good BOVs for the partial implementation process.

Besides, it is necessary to go ahead in the design of a *benchmark* that takes into account the different aspects and nature of these problem and algorithms in order to direct us searching the best strategies.

Acknowledgements

Supported by grant TIN11-28084 of the Science and Innovation Ministry of Spain.

References

[Bizer et al., 2009] Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked data-the story so far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3):1–22.

[Codd, 1970] Codd, E. F. (1970). A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387.

[Cordero et al., 2013] Cordero, P., Enciso, M., and Mora, A. (2013). Automated reasoning to infer all minimal keys. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI'13, pages 817–823. AAAI Press.

[Cordero et al., 2014] Cordero, P., Enciso, M., Mora, A., and de Guzmán, I. P. (2014). A tableaux-like method to infer all minimal keys. *Logic Journal of the IGPL*, 22(6):1019–1044.

[Elmasri and Navathe, 2010] Elmasri, R. and Navathe, S. (2010). *Fundamentals of Database Systems*. Prentice Hall International, 6 edition.

[Fayyad et al., 1996] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, pages 37–54.

[Huhtala et al., 1999] Huhtala, Y., Krkkinen, J., Porkka, P., and Toivonen, H. (1999). Tane: An efficient algorithm for discovering functional and approximate dependencies. *Comput. J.*, 42(2):100–111.

[Kemper and Moerkotte, 1991] Kemper, A. and Moerkotte, G. (1991). Query optimization in object bases: Exploiting relational techniques. In *Query Processing for Advanced Database Systems, Dagstuhl*, pages 63–98. Morgan Kaufmann.

[Manolopoulos et al., 1999] Manolopoulos, Y., Theodoridis, Y., and Tsotras, V. J. (1999). *Advanced Database Indexing*, volume 17 of *Advances in Database Systems*. Kluwer.

[Morgan, 1992] Morgan, C. G. (1992). An automated theorem prover for relational logic (abstract). In Fronhfer, B., Hhnle, R., and Kufl, T., editors, *TABLEAUX*, pages 56–58.

[Risch and Schwind, 1992] Risch, V. and Schwind, C. (1992). Tableaux-based theorem proving and non-standard reasoning. In Fronhfer, B., Hhnle, R., and Kufl, T., editors, *TABLEAUX*, pages 76–78.

[Simsion and Witt, 2005] Simsion, G. C. and Witt, G. C. (2005). *Data modeling essentials*. Amsterdam; Boston, 3rd edition.

[Sismanis et al., 2006] Sismanis, Y., Brown, P., Haas, P. J., and Reinwald, B. (2006). Gordian: efficient and scalable discovery of composite keys. In *In Proc. International Conference on Very Large Data Bases (VLDB*, pages 691–702.

[Witten et al., 2011] Witten, I., Frank, E., and Hall, M. (2011). *Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

 ISBN: 978-84-617-2230-3

[Yao et al., 2002] Yao, H., Hamilton, H. J., and Butz, C. J. (2002). Fdmine: Discovering functional dependencies in a database using equivalences. In *ICDM*, pages 729–732. IEEE Computer Society.

 ISBN: 978-84-617-2230-3

Apéndice D

Increasing the Efficiency of Minimal Key Enumeration Methods by Means of Parallelism

Increasing the Efficiency of Minimal Key Enumeration Methods by Means of Parallelism

Fernando Benito[1], Pablo Cordero[2], Manuel Enciso[1] and Ángel Mora[2]
[1]*Department of Languages and Computer Science, University of Málaga, Málaga, Spain*
[2]*Department of Applied Mathematics, University of Málaga, Málaga, Spain*

Keywords: Functional Dependency, Minimal Key, Parallelism, Logic, Tableaux-method.

Abstract: Finding all minimal keys in a table is a hard problem but also provides a lot of benefits in database design and optimization. Some of the methods proposed in the literature are based on logic and, more specifically on tableaux paradigm. The size of the problems such methods deal with is strongly limited, which implies that they cannot be applied to big database schemas. We have carried out an experimental analysis to compare the results obtained by these methods in order to estimate their limits. Although tableaux paradigm may be viewed as a search space guiding the key finding task, none of the previous algorithms have incorporated parallelism. In this work, we have developed two different versions of the algorithms, a sequential and a parallel one, stating clearly how parallelism could naturally be integrated and the benefits we get over efficiency. This work has also guided future work guidelines to improve future designs of these methods.

1 INTRODUCTION

Identifying properly all the keys of a table in a relational schema is a crucial task for several areas in information management: data modeling (Simsion and Witt, 2005), query optimization (Kemper and Moerkotte, 1991), indexing (Manolopoulos et al., 1999), etc. Key constraints specify sets of attributes in a relation such that their projection univocally identifies each tuple of the relation. Therefore, each key is composed by a subset of attributes playing the role of a *domain* in a given function whose *image* is the whole set of attributes. This way, the table is viewed as its extensional definition. These functions are described by means of a *Functional Dependency (FD)* which specifies a constraint between two subset of attributes, denoted $A \rightarrow B$, ensuring us that for any two tuples in a table, if they agree on A, they also agree on B.

All functional dependencies satisfied in a given table may be deduced from its dataset using data mining techniques (Appice et al., 2011; Huhtala et al., 1999), or may be provided by database designers. It is out the scope of this work to extract FDs from relational tables, since it becomes a data mining problem (Fayyad et al., 1996). Minimal key problem consists in finding all the attribute subsets which make up a minimal key given a set of FDs occurring within a schema of a relational table.

Nowadays, several algorithms are capable to solve this problem using different classical techniques (Lucchesi and Osborn, 1978; Yu and Johnson, 1976; Saiedian and Spencer, 1996; Zhang, 2009; Armstrong, 1974) (see Section 2 for further details). Recently, some alternative methods have been introduced using logic. In this work we will concentrate on algorithms guided by logic, and most specifically, those using tableaux paradigm (Morgan, 1992) for deriving keys of a relation schema using inference systems. As we shall see later, tableaux might be considered a flexible and powerful framework to design automated deduction methods to solve complex problems in a flexible and effective way.

In previous works, several tableaux-like methods have been introduced (Wastl, 1998a; Cordero et al., 2013). Efficient versions of both of them have been implemented. Nevertheless, tableaux-like methods generate wide search spaces and, in many cases, the same solution (same minimal key) appears at the end of several branches of the tree representing the search space. These intrinsic characteristics of tableaux-like methods establish a strong limitation in the size of the problems to be treated by them and, usually, discourage their use.

Indeed, sequential implementation of these methods produces such an explosion of search space that we go beyond machine capabilities, even with small

Benito Picazo F., Cordero P., Enciso M. and Mora Á..
Increasing the Efficiency of Minimal Key Enumeration Methods by Means of Parallelism.
DOI: 10.5220/0005108205120517
In *Proceedings of the 9th International Conference on Software Engineering and Applications* (ICSOFT-EA-2014), pages 512-517
ISBN: 978-989-758-036-9

problems (20+ FDs for $\mathbb{K}$'s method (Wastl, 1998a) as it has been demonstrated in previous studies of this work (Cordero et al., 2013)). However, a very interesting property within search spaces induced by tableaux methods, is the fully independence of their branches, so we can directly consider them as separated sub-problems leading to several solutions of the original problem. It is in this spirit that tableaux paradigm supplies us an optimal guide to build parallel algorithms finding all minimal keys in a table, since the building of the tableaux tree directs the parallel and independent processing.

In this work, we have developed parallel implementations of tableaux-like methods to solve minimal keys finding problems. As shown in the following, they have significantly increased the size of the problem these methods are able to handle. Intentionally, we have executed them under different hardware configurations trying to discover tendencies in which the efficiency within method could be influenced. As already implied above, a processing cluster will be available for us to engage problems with a substantial size at the input so we can deal with them in terms of storage capacity and execution time.

The paper is organized as follows: In Section 2 we introduce several background. A brief study of the state-of-the-art is exposed in Section 3. Section 4 introduces us into sequential tableaux-like methods showing some experimental results. Parallelism implementations enter the scene in Section 5 presenting their way to proceed, and showing the high benefits obtained. Several conclusions are then given in Section 6.

2 BACKGROUND

We begin this section with three brief definitions of basic concepts.

Definition 1 (Functional dependency)**.** *Let U be a set of attributes. A functional dependency (FD) over U is an expression of the form $X \rightarrow Y$, where X,Y are attribute sets. It is satisfied in a table R if for every two tuples of R, if they agree on X, then they agree on Y.*

A key of a relational table is a subset of attributes that allows us to uniquely characterize each row. It may be defined by means of functional dependencies as follows:

Definition 2 (Key)**.** *Given a table R over the set of attributes U, we say that K is a key in R if the functional dependency $K \rightarrow U$ holds in R.*

Definition 3 (Minimal Key)**.** *Given the table R, the attribute set $K \subset U$ is said to be a minimal key if it is a key of R and for all attribute $k \in K$ the subset $K - \{k\}$ is not a key of R.*

Due to space limitation, we refer those readers non familiar with the formal notions of FDs, keys and relational tables to (Elmasri and Navathe, 2010). In Table 1, we just illustrate its semantics by a basic example.

From the information in Table 1, we may ensure that the following FDs are satisfied: $Title,Year \rightarrow Country$, $Title,Year \rightarrow Director$ and $Director \rightarrow Nationality$. Moreover, the table has only one minimal key: $\{Title,Year,Star\}$

Inferring minimal keys from a set of FDs has been well studied. The algorithm of Lucchesi and Osborn (Lucchesi and Osborn, 1978) to find all the keys in a relational schema is considered the first deep study around this problem. Yu and Johnson (Yu and Johnson, 1976) established that the number of keys is limited by the factorial of the number of dependencies, so, there does not exist a polynomial algorithm for this problem.

3 TABLEAUX-LIKE METHODS

Some authors have used several techniques to solve this problem. Saiedian and Spencer (Saiedian and Spencer, 1996) propose an algorithm for computing the candidate keys using attribute graphs. Zhang in (Zhang, 2009) uses Karnaugh maps to calculate all the keys. Nevertheless, we are interested in the use of artificial intelligence techniques and, more specifically, in the use of logic. Armstrong's axiomatic system (Armstrong, 1974) is the former system introduced to manage FDs in a logic style. In (Wastl, 1998b), for the first time a Hilbert-styled inference system for deriving all keys of a relation schema was introduced. Alternatively, in (Cordero et al., 2013) the authors tackle the key finding problem with another inference rule inspired by the Simplification Logic for Functional Dependencies. These two papers constitute the target algorithms to be compared in this work. We refer the reader to the original papers for further details.

Both works are strongly based on tableaux paradigm. Tableaux-like methods represent the search space as a tree, where its leaves contain the solutions (keys). Tree building process begins with an initial root and from there on, inference rules generate new branches labeled with nodes representing simpler instances of the parent node. The very best advantage of this process goes to its versatility, since developing new inference systems –which is not a trivial task indeed– allows us to design a new method. Com-

Table 1: Movie table.

Title	Year	Country	Director	Nationality	Star
Pulp Fiction	1994	USA	Quentin Tarantino	USA	John Travolta
Pulp Fiction	1994	USA	Quentin Tarantino	USA	Uma Thurman
Pulp Fiction	1994	USA	Quentin Tarantino	USA	Samuel L. Jackson
King Kong	2005	New Zealand	Peter Jackson	New Zealand	Naomi Watts
King Kong	2005	New Zealand	Peter Jackson	New Zealand	Jack Black
King Kong	1976	USA	De Laurentiis	IT	Jessica Lange
King Kong	1976	USA	De Laurentiis	IT	Jeff Bridges
Django Unchained	2012	USA	Quentin Tarantino	USA	Jamie Foxx
Django Unchained	2012	USA	Quentin Tarantino	USA	Samuel L. Jackson

parisons between tableaux-like methods can be easily drawn as its efficiency goes hand-in-hand with the size of the generated tree.

Although $\mathbb{K}$ and SL_{FD} are the two inference systems which are the basis of two tableaux-like methods, they are very different. $\mathbb{K}$ is the former basis for a Hilbert-styled method and it deals with unitary functional dependencies, which produce a significant growth of the input set. SL_{FD} avoids the use of fragmentation by using generalized formulas. It is guided by the idea of simplifying the set of FDs by removing redundant attributes efficiently. Moreover, SL_{FD} incorporates a pre-processing task at a first step which prunes the input up to an algebraic characterization of the problem by providing significant reduction of the problem's size to be treated by the tableaux-like method, which is the hardest task.

4 LIMITS OF SEQUENTIAL TABLEAUX-LIKE METHODS

As we mentioned in the introduction, in this section we show the strong limitation that sequential implementation of tableaux-like methods will face to solve minimal keys.

We have developed two efficient implementations of both methods and they have been executed over a high performance architecture sited in the Supercomputing and Bioinnovation Center [1]. In these experiments we take into account the following parameters: execution time in seconds (named [Ti]), number of nodes in the tableaux search space (named [No]) and number of redundant keys (named [RK]). This last parameter shows the impact of extra branches, i.e., the number of duplicated keys computed by the algorithm.

Execution times may be considered a parameter linked to the architecture we are using for running the experiments. Number of nodes and redundant keys represent the size of the search space and the repeated operations respectively. They are independent of the implementation, providing a fair comparison between methods in the future.

[1] http://www.scbi.uma.es.

4.1 First Experiment: Benchmarking Problems

Although there not exists a benchmarking for functional dependency problems, our first intention was to execute the methods over a set of different and characteristic problems. Thus, we began building a battery of problems gathered from several related papers around (Saiedian and Spencer, 1996; Wastl, 1998a) conforming an initial suitable set of problems in a benchmarking style. Results obtained for this battery of problems are shown in Table 2.

As shown in Table 2 only one of the entry problems needs more than one second to finalize and it is in the case of $\mathbb{K}$ method. This is due to the small sets handled by these academic problems. Results for SL_{FD} are better than $\mathbb{K}$ except for saiedian3 problem. Indeed, there are cases where we need just one node to finish the algorithm, corresponding to those problems where the algebraic characterization used by SL_{FD} reduces the problem to its canonical version.

4.2 Second Experiment: Random Problems

In this subsection we deal with larger randomly generated problems. We have built several examples varying two parameters: number of attributes and number of FDs. We have not established a correlation between both parameters in the generation because different ratios between them produce problems with significant differences. Moreover, observe that number of minimal keys is not directly influenced by these two parameters.

The size of the problems in Table 3 is greater than those presented in previous section. They may be considered *medium*-size problems, having param-

Table 2: Sequential executions over benchmarking problems.

Problem	Attrib	FDs	Keys	$\mathbb{K}$			SL_{FD}		
				[Ti]	[No]	[RK]	[Ti]	[No]	[RK]
saiedian1	6	5	1	0	12	5	0	1	0
saiedian2	6	5	3	0	7	0	0	10	3
saiedian3	7	7	4	0	139	64	0	674	284
derivation5	9	4	5	0	17	4	0	41	20
a3rojo	7	5	2	0	81	29	0	4	0
elmasri1	6	3	1	0	1	0	0	1	0
wastl2	7	3	1	0	2	0	0	1	0
wastl10	3	3	1	0	5	2	0	1	0
wastl13	4	4	3	0	10	2	0	13	5
example001	10	7	8	20	55.768	24.174	0	1.090	448

Table 3: Sequential executions over random problems.

Problem	Attrib	FDs	Keys	$\mathbb{K}$			SL_{FD}		
				[Ti]	[No]	[RK]	[Ti]	[No]	[RK]
med1	10	10	3	155	1.463.228	723.372	0	902	404
med2	5	17	4	0	1.906	1.029	0	1.149	772
med3	15	7	2	40	432.104	220.230	0	143	71
med4	30	5	5	12	802.770	300.485	12	23	7
med5	20	10	3	180	2.038.746	1.012.651	3	1.102	666
med6	5	50	5	20	218.892	179.444	1	129.508	117.461
med7	40	10	2	204	1.130.539	467.512	0	122	53
med8	15	15	7	38	6.715.949	3.023.693	5	77.922	41.070
med9	7	50	5	325	32.219.336	21.357.930	18	2.760.961	2.227.596
med10	10	20	4	835	496.380.119	218.275.528	8	20.442	9.966

eters which properly match with real tables in software engineering and execution times are quite reasonable, particularly for SL_{FD} method (less than one minute). Nevertheless, we notice that as far as problem's size grows (even just a little), results go considerably beyond. Therefore, we are absolutely limited when dealing with real problems, where the input size would be worthy of consideration. So the challenge is to figure out whether parallelism will help us solving these kind of problems, and even greater ones.

5 PARALLELIZATION OF TABLEAUX-LIKE METHODS

As a conclusion of the experiments presented in previous section, parallel strategy and big hardware resources will be totally indispensable if we want to compare tableaux-like methods from one to another. Our intention is to take advantage of tableaux design to split the original problem into *atomic* instances that may be solved within a reasonable time and resources. Then, we combine the solutions of all these sub-problems to enumerate all the minimal keys.

Thus, our parallel implementations of the algorithms will work in two steps:

1. Splitting step: It executes the tableaux-like methods but it will stop in a determinate level that we will introduce at the process call, generating a set of sub-problems. The level is induced by the size of the root (the number of attributes in this level).
2. Parallel step: In this second stage we execute parallel task solving those sub-problems and, at the end, we combine all the solutions to get all minimal keys.

In order to test parallel versions we run another battery of problems whose results are retrieved in Table 4. This time we include several new columns gathering parallel implementation's parameters: Break-off value [L] and number of generated sub-problems [Sp].

It is imperative to state some critical considerations concerning the limit size of the atomic problems. In one hand, we have observed that the greater this limit is, the better will be the improvement by parallelism, since it *would* generate a higher number of sub-problems. However, as far as we try to make this improvement to be better by a wide limit value, the longer the partial version will take to split the entry problem.

In addition, this is not an independent parameter among the algorithms, we need to choose a different break-off value depending on the method we are using as each one of them will need a particular amount of resources.

Table 4: Parallel executions over random problems.

Prob	Attr	FDs	Keys	$\mathbb{K}$					SL_{FD}				
				[L]	[Sp]	[Ti]	[No]	[RK]	[L]	[Sp]	[Ti]	[No]	[RK]
cp01	7	50	5	6	87	0	32.219.336	21.357.930	45	50	3	2.760.961	2.227.596
cp02	10	20	4	9	44	0	496.380.119	218.275.528	10	268	11	20.442	9.966
cp03	15	20	3	11	32	1	17.917.662	9.340.225	15	78	4	1.405.153	814.026
cp04	20	20	4	10	27.284	31	3.145.751.761	1.424.991.475	15	47	3	587.765	313.513
cp05	20	30	45	12	1.696	3	1.563.813.853	677.457.455	24	98	136	271.402.277	162.828.760
cp06	10	35	5	8	2.358	3	121.396.806	65.571.971	30	158	8	3.215.995	1.686.149
cp07	25	15	15	14	25.836	28	3.975.400.144	1.980.982.101	8	802	39	220.047	914.80
cp08	15	40	1	9	39.708	46	837.341.359	433.068.418	0	0	1	1	0
cp09	25	20	25	14	33.146	38	123.283.772.804	59.975.556.886	12	18.999	1.077	47.014.652	23.418.562
cp10	35	10	37	22	1.370	5	101.429.265.443	68.138.197.993	7	219	10	27.985	13.436

As an example of this last point, we would like to check execution times for cp09 and cp10 in the case of $\mathbb{K}$, where the huge difference is due to a wiser splitting process.

As a general conclusion, execution times are pretty reasonable considering the dimension of these entry problems (several minutes were enough to resolve most of cases).

On a separate issue, we can notice that results are pretty huge for this kind of problems using $\mathbb{K}$'s method. The hugest number of nodes of the tableaux overtakes up to 123 billions of nodes. So, efficiency of $\mathbb{K}$'s method is so far to be accepted. SL_{FD} reaches better times and dimensions tableaux in the very most of cases. Thus, several noteworthy outcomes come up.

First, problem cp04 needs a tableaux of over half-million nodes with SL_{FD}, while $\mathbb{K}$ goes up to 3 billions! This is due to the high number of FDs after the hard fragmentation rule inherent to $\mathbb{K}$. A similar situation involves cp09 and cp10 problems.

Finally, if we care about useless computing time, we notice the number of redundant keys is terrible; the difference here between both methods is as impressive as in the rest of parameters. For instance, the resulting set of minimal keys for cpx04 problem contains just 4 minimal keys. However, $\mathbb{K}$ generates 1,424,991,475 redundant keys and SL_{FD} 313,513. This is indeed, a huge waste of space and time.

6 CONCLUSIONS

The first point we want to state clear is that the concept of parallelism we are dealing with refers to a *hardware parallelism*. We mean that the benefits we are obtained from parallelism are due to a cluster of cores where we can deliver each of our jobs continuously.

In order to resolve real problems where the size of the input is substantial, it is imperative to count on the participation of a great amount of resources. Moreover, it is difficult, at a first look, to estimate whether a given problem will result in a difficult or an easy one. In some sense we may say that it is a chaotic and unpredictable problem.

A glance is enough to easily realize that $\mathbb{K}$ algorithm needs more time, more nodes and more redundant keys to reach the solution than SL_{FD} in the very most of cases. Indeed, the differences are not trivial so far. Concerning the size of the tableaux, $\mathbb{K}$ builds up billion of nodes whereas SL_{FD} generates a reasonable amount of nodes. A similar conclusion may be established for the number of redundant keys.

Finally, establishing an appropriate limit to split up the entry problem in the parallel versions of the algorithms is not an easy issue so far. We have to run several experiments to reach a good value which will not spend so much time splitting the entry but it should spend time enough to take advantage of the parallelism.

ACKNOWLEDGEMENTS

Supported by Grant TIN2011-28084 of the Science and Innovation Ministry of Spain.

REFERENCES

Appice, A., Ceci, M., Turi, A., and Malerba, D. (2011). A parallel, distributed algorithm for relational frequent pattern discovery from very large data sets. *Intell. Data Anal.*, 15(1):69–88.

Armstrong, W. W. (1974). Dependency structures of data base relationships. In *IFIP Congress*, pages 580–583.

Cordero, P., Enciso, M., and Mora, A. (2013). Automated reasoning to infer all minimal keys. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI'13, pages 817–823. AAAI Press.

Elmasri, R. and Navathe, S. (2010). *Fundamentals of Database Systems*. Prentice Hall International, 6 edition.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, pages 37–54.

Huhtala, Y., Krkkinen, J., Porkka, P., and Toivonen, H. (1999). Tane: An efficient algorithm for discovering functional and approximate dependencies. *Comput. J.*, 42(2):100–111.

Kemper, A. and Moerkotte, G. (1991). Query optimization in object bases: Exploiting relational techniques. In *Query Processing for Advanced Database Systems, Dagstuhl*, pages 63–98. Morgan Kaufmann.

Lucchesi, C. L. and Osborn, S. L. (1978). Candidate keys for relations. *J. Comput. Syst. Sci.*, 17(2):270–279.

Manolopoulos, Y., Theodoridis, Y., and Tsotras, V. J. (1999). *Advanced Database Indexing*, volume 17 of *Advances in Database Systems*. Kluwer.

Morgan, C. G. (1992). An automated theorem prover for relational logic (abstract). In Fronhfer, B., Hhnle, R., and Kufl, T., editors, *TABLEAUX*, pages 56–58.

Saiedian, H. and Spencer, T. (1996). An efficient algorithm to compute the candidate keys of a relational database schema. *Comput. J.*, 39(2):124–132.

Simsion, G. C. and Witt, G. C. (2005). *Data modeling essentials*. Amsterdam; Boston, 3rd edition.

Wastl, R. (1998a). Linear derivations for keys of a database relation schema. *J. UCS*, 4(11):883–897.

Wastl, R. (1998b). On the number of keys of a relational database schema. *Journal of Universal Computer Science*, 4.

Yu, C. T. and Johnson, D. T. (1976). On the complexity of finding the set of candidate keys for a given set of functional dependencies. *Inf. Process. Lett.*, 5(4):100–101.

Zhang, Y. (2009). Determining all candidate keys based on karnaugh map. *IEEE International Conference on Information Management, Innovation Management and Industrial Engineering*, 04:226–229.

Bibliografía

[1] Adomavicius, G., and Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng. 17*, 6 (June 2005), 734–749.

[2] Adomavicius, G., and Tuzhilin, A. In *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer, 2011, pp. 217–253.

[3] Al-Ali, R., Kathiresan, N., Anbari, M. E., Schendel, E. R., and Zaid, T. A. Workflow optimization of performance and quality of service for bioinformatics application in high performance computing. *Journal of Computational Science 15* (2016), 3 – 10. International Computational Science and Engineering Conference 2015 (ICSEC15).

[4] Armstrong, W. W. Dependency structures of data base relationships. In *Proceedings of the International Federation for Information Processing Congress* (1974), pp. 580–583.

[5] Armstrong, W. W., and Deobel, C. Decompositions and functional dependencies in relations. *ACM Trans. Database Syst. 5*, 4 (Dec. 1980), 404–430.

[6] Atzeni, P., and De Antonellis, V. *Relational Database Theory.* Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1993.

[7] Beeri, C., and Vardi, M. Y. The implication problem for data dependencies. In *Automata, Languages and Programming* (Berlin, Heidelberg, 1981), S. Even and O. Kariv, Eds., Springer Berlin Heidelberg, pp. 73–85.

[8] Belohlavek, R. Similarity relations in concept lattices. *J. Log. Comput. 10*, 6 (2000), 823–845.

[9] Benito-Picazo, F. *Minimal Key-Par. Una versión paralela de los algoritmos de búsqueda de claves minimales basados en Tableaux.* Proyecto Fin de Carrera. Dpto. Lenguajes y Ciencias de la Computación, Universidad de Málaga, 2013.

[10] Benito-Picazo, F. *Study of minimal keys algorithms based on tableaux methods. Increasing the range of treated problems by means of parallelism and prune strategies based on minimal subsets.* Master Thesis. Languages and Computer Science Department, Universidad de Málaga, 2014.

[11] Benito-Picazo, F., Cordero, P., Enciso, M., and Mora, A. Increasing the efficiency of minimal key enumeration methods by means of parallelism. In *ICSOFT-EA 2014 - Proceedings of the 9th International Conference on Software Engineering and Applications, Vienna, Austria, 29-31 August, 2014* (2014), pp. 512–517.

[12] Benito-Picazo, F., Cordero, P., Enciso, M., and Mora, A. Closed sets enumeration: a logical approach. In *Proceedings of the Seventeenth International Conference on Computational and Mathematical Methods in Science and Engineering* (Cádiz, Spain, 2017), pp. 287–292.

[13] Benito-Picazo, F., Cordero, P., Enciso, M., and Mora, A. Reducing the search space by closure and simplification paradigms. *Journal of Supercomputing 73*, 1 (Jan. 2017), 75–87.

[14] Benito-Picazo, F., Cordero, P., Enciso, M., and Mora, A. Minimal generators, an affordable approach by means of massive computation. *The Journal of Supercomputing* (Online Jun 2018).

[15] Benito-Picazo, F., Enciso, M., Rossi, C., and Guevara, A. Enhancing the conversational process by using a logical closure operator in phenotypes implications. *Mathematical Methods in the Applied Sciences 41*, 3 (2017), 1089–1100.

[16] Bertet, K., Demko, C., Viaud, J.-F., and Guérin, C. Lattices, closures systems and implication bases: A survey of structural aspects and algorithms. *Theoretical Computer Science 743* (2018), 93 – 109.

[17] Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. Recommender systems survey. *Knowledge-Based Systems 46* (2013), 109 – 132.

[18] Burusco, A., and Fuentes-González, R. Construction of the l-fuzzy concept lattice. *Fuzzy Sets and Systems 97*, 1 (1998), 109–114.

[19] Cano, E., and Morisio, M. Characterization of public datasets for recommender systems. In *2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)* (Sept 2015), pp. 249–257.

[20] Chen, L., and Pu, P. Hybrid Critiquing-based Recommender Systems. In *Proceedings of the 12th International Conference on Intelligent User Interfaces* (New York, NY, USA, 2007), IUI '07, ACM, pp. 22–31.

[21] Codd, E. F. A relational model of data for large shared data banks. *Commun. ACM 13*, 6 (1970), 377–387.

[22] Codd, E. F. Further normalization of the data base relational model. *IBM Research Report, San Jose, California RJ909* (1971).

[23] Codd, E. F. Normalized data base structure: A brief tutorial. In *Proceedings of the 1971 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control* (New York, NY, USA, 1971), SIGFIDET '71, ACM, pp. 1–17.

[24] Codd, E. F. *The Relational Model for Database Management: Version 2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[25] Cordero, P., Enciso, M., and Mora, A. Automated reasoning to infer all minimal keys. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence* (2013), IJCAI'13, AAAI Press, pp. 817–823.

[26] Cordero, P., Enciso, M., Mora, A., and de Guzmán, I. P. SLFD logic: Elimination of data redundancy in knowledge representation. In *IBERAMIA 2002: Proceedings of the 8th Ibero-American Conference on AI* (London, UK, 2002), Springer-Verlag, pp. 141–150.

[27] Cordero, P., Enciso, M., Mora, A., and de Guzmán, I. P. A tableaux-like method to infer all minimal keys. *Logic Journal of the IGPL 22*, 6 (2014), 1019–1044.

[28] Cordero, P., Enciso, M., Mora, A., and Ojeda-Aciego, M. Computing minimal generators from implications: a logic-guided approach. In *Proceedings of The Ninth International Conference on Concept Lattices and Their Applications, Fuengirola (Málaga), Spain, October 11-14, 2012* (2012), pp. 187–198.

[29] Cordero, P., Enciso, M., Mora, A., Ojeda-Aciego, M., and Rossi, C. Knowledge discovery in social networks by using a logic-based treatment of implications. *Knowledge-Based System 87* (2015), 16–25.

[30] Costante, E., den Hartog, J., Petkovic, M., Etalle, S., and Pechenizkiy, M. A white-box anomaly-based framework for database leakage detection. *Journal of Information Security and Applications 32* (2017), 27 – 46.

[31] Crespo, R. G., Martínez, O. S., Lovelle, J. M. C., García-Bustelo, B. C. P., Gayo, J. E. L., and de Pablos, P. O. Recommendation system based on user interaction data applied to intelligent electronic books. *Computers in Human Behavior 27*, 4 (2011), 1445 – 1449. Social and Humanistic Computing for the Knowledge Society.

[32] Darwen, H., Date, C. J., and Fagin, R. A normal form for preventing redundant tuples in relational databases. In *Proceedings of the 15th International Conference on Database Theory* (New York, NY, USA, 2012), ICDT '12, ACM, pp. 114–126.

[33] de Campos, L. M., Fernández-Luna, J. M., Huete, J. F., and Rueda-Morales, M. A. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *International Journal of Approximate Reasoning 51*, 7 (2010), 785 – 799.

[34] de Moraes, N. R. M., Dias, S. M., Freitas, H. C., and Zárate, L. E. Parallelization of the next closure algorithm for generating the minimum set of implication rules. *Artificial Intelligence Research 5* (03 2016), 40–54.

[35] Dean, J., and Ghemawat, S. Mapreduce: Simplified data processing on large clusters. *Commun. ACM 51*, 1 (Jan. 2008), 107–113.

[36] du Boucher-Ryan, P., and Bridge, D. Collaborative recommending using formal concept analysis. *Knowledge-Based Systems 19*, 5 (2006), 309 – 315.

[37] Eirinaki, M., Gao, J., Varlamis, I., and Tserpes, K. Recommender systems for large-scale social networks: A review of challenges and solutions. *Future Generation Computer Systems 78* (2018), 413 – 418.

[38] Elmasri, R., and Navathe, S. *Fundamentals of Database Systems*, 6 ed. Prentice Hall International, 2010.

[39] Endres, D., Adam, R., Giese, M. A., and Noppeney, U. Understanding the semantic structure of human fmri brain recordings with formal concept analysis. In *Proceedings of the 10th International Conference on Formal Concept Analysis* (Berlin, Heidelberg, 2012), ICFCA'12, Springer-Verlag, pp. 96–111.

[40] Fadous, R., and Forsyth, J. Finding candidate keys for relational data bases. In *SIGMOD Conference* (1975), W. F. King, Ed., ACM, pp. 203–210.

[41] Fagin, R. A normal form for relational databases that is based on domains and keys. *ACM Trans. Database Syst. 6*, 3 (Sept. 1981), 387–415.

[42] Feil, S., Kretzer, M., Werder, K., and Maedche, A. Using gamification to tackle the cold-start problem in recommender systems. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion* (New York, NY, USA, 2016), CSCW '16 Companion, ACM, pp. 253–256.

[43] Feng, H., Tian, J., Wang, H. J., and Li, M. Personalized recommendations based on time-weighted overlapping community detection. *Information & Management 52*, 7 (2015), 789–800.

[44] Friedman, A., Knijnenburg, B. P., Vanhecke, K., Martens, L., and Berkovsky, S. *Privacy Aspects of Recommender Systems*. Springer US, Boston, MA, 2015, pp. 649–688.

[45] Ganter, B., Stumme, G., and Wille, R. *Formal concept analysis: Methods and applications in computer science*. TU Dresden, 2002, pp. 1–85.

[46] Ganter, B., and Wille, R. *Formal Concept Analysis: Mathematical Foundations*, 1st ed. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[47] Gavalas, D., Konstantopoulos, C., Mastakas, K., and Pantziou, G. Mobile recommender systems in tourism. *Journal of Network and Computer Applications 39* (2014), 319 – 333.

[48] Goh, T. N. An information management paradigm for statistical experiments. *Quality and Reliability Eng. Int. 26*, 5 (2010), 487–494.

[49] Gras, B., Brun, A., and Boyer, A. Identifying grey sheep users in collaborative filtering: A distribution-based technique. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization* (New York, NY, USA, 2016), UMAP '16, ACM, pp. 17–26.

[50] Griol, D., and Molina, J. M. Building multi-domain conversational systems from single domain resources. *Neurocomputing 271* (2018), 59 – 69.

[51] Gunawardana, A., and Shani, G. A survey of accuracy evaluation metrics of recommendation tasks. *J. Mach. Learn. Res. 10* (Dec. 2009), 2935–2962.

[52] Gunawardana, A., and Shani, G. *Evaluating Recommender Systems*. Springer US, Boston, MA, 2015, pp. 265–308.

[53] Guo, G. Resolving data sparsity and cold start in recommender systems. In *Proceedings of the 20th International Conference on User Modeling, Adaptation, and Personalization* (Berlin, Heidelberg, 2012), UMAP'12, Springer-Verlag, pp. 361–364.

[54] Harper, F. M., and Konstan, J. A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst. 5*, 4 (Jan. 2016), 19:1–19:19.

[55] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst. 22*, 1 (Jan. 2004), 5–53.

[56] Hill, W., Stead, L., Rosenstein, M., and Furnas, G. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1995), CHI '95, ACM Press/Addison-Wesley Publishing Co., pp. 194–201.

[57] Huhtala, Y., Karkkainen, J., Porkka, P., and Toivonen, H. Tane: An efficient algorithm for discovering functional and approximate dependencies. *Comput. J. 42*, 2 (1999), 100–111.

[58] Ibaraki, T., Kogan, A., and Makino, K. Functional dependencies in horn theories. *Artificial Intelligence 108*, 1 (1999), 1 – 30.

[59] Isinkaye, F., Folajimi, Y., and Ojokoh, B. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal 16*, 3 (2015), 261 – 273.

[60] Jannach, D., Zanker, M., and Fuchs, M. Constraint-Based Recommendation in Tourism: A Multiperspective Case Study. *Information Technology & Tourism 11*, 2 (2009), 139–155.

[61] Kamra, A., Terzi, E., and Bertino, E. Detecting anomalous access patterns in relational databases. *The VLDB Journal 17*, 5 (Aug. 2008), 1063–1077.

[62] Karnaugh, M. The map method for synthesis of combinational logic circuits. *American Institute of Electrical Engineers, Part I: Communication and Electronics, Transactions of the 72*, 5 (Nov. 1953), 593–599.

[63] Kemper, A., and Moerkotte, G. Query optimization in object bases: Exploiting relational techniques. In *Query Processing for Advanced Database Systems, Dagstuhl*. Morgan Kaufmann, 1991, pp. 63–98.

[64] Kent, W. A simple guide to five normal forms in relational database theory. *Commun. ACM 26*, 2 (Feb. 1983), 120–125.

[65] Kester, Q.-A. Application of formal concept analysis to visualization of the evaluation of risks matrix in software engineering projects. *International Journal of Science, Engineering and Technology Research (IJSETR) 2* (Jan 2013), 220–225.

[66] Kim, K. An improved semi-supervised dimensionality reduction using feature weighting: Application to sentiment analysis. *Expert Systems with Applications 109* (2018), 49 – 65.

[67] Kuznetsov, S., and Obiedkov, S. Comparing performance of algorithms for generating concept lattices. *Journal of Experimental and Theoretical Artificial Intelligence 14* (2002), 189–216.

[68] Lampropoulos, A. S., Lampropoulou, P. S., and Tsihrintzis, G. A. A cascade-hybrid music recommender system for mobile services based on musical genre classification and personality diagnosis. *Multimedia Tools Appl. 59*, 1 (2012), 241–258.

[69] Lee, S., and Choi, J. Enhancing user experience with conversational agent for movie recommendation: Effects of self-disclosure and reciprocity. *International Journal of Human-Computer Studies 103* (2017), 95 – 105.

[70] Leiva, J. L., Enciso, M., Rossi, C., Cordero, P., Mora, A., and Guevara, A. Context-aware recommendation using fuzzy formal concept analysis. In *ICSOFT* (2013), J. Cordeiro, D. A. Marca, and M. van Sinderen, Eds., SciTePress, pp. 617–623.

[71] Leiva, J. L., Enciso, M., Rossi, C., Cordero, P., Mora, A., and Guevara, A. Improving recommender systems with simplification logic to manage implications with grades. In *Software Technologies - 8th International Joint Conference, ICSOFT 2013, Reykjavik, Iceland, July 29-31, 2013, Revised Selected Papers* (2013), pp. 290–305.

[72] Linden, G., Smith, B., and York, J. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing 7* (2003), 76–80.

[73] Lops, P., de Gemmis, M., and Semeraro, G. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer, 2011, pp. 73–105.

[74] Lucchesi, C. L., and Osborn, S. L. Candidate keys for relations. *J. Comput. Syst. Sci. 17*, 2 (1978), 270–279.

[75] Maier, D. *The Theory of Relational Databases.* Computer Science Press, Rockville, 1983.

[76] Maier, D., Mendelzon, A. O., and Sagiv, Y. Testing implications of data dependencies. *ACM Trans. Database Syst. 4*, 4 (Dec. 1979), 455–469.

[77] Mandl, M., Felfernig, A., Teppan, E., and Schubert, M. Consumer decision making in knowledge-based recommendation. *Journal of Intelligent Information Systems 37* (2011), 1–22.

[78] Manolopoulos, Y., Theodoridis, Y., and Tsotras, V. J. *Advanced Database Indexing*, vol. 17 of *Advances in Database Systems.* Kluwer, 1999.

[79] McEneaney, W. M., Deshpande, A., and Gaubert, S. Curse-of-complexity attenuation in the curse-of-dimensionality-free method

for hjb pdes. In *2008 American Control Conference* (June 2008), pp. 4684–4690.

[80] McSherry, D. Minimizing dialog length in interactive case-based reasoning. In *Procs of the 17th Int Joint Conf on AI, IJCAI, Seattle, Washington, USA, August 4-10, 2001* (2001), pp. 993–998.

[81] Medina-Moreira, J., Apolinario, O., Luna-Aveiga, H., Lagos-Ortiz, K., Paredes-Valverde, M. A., and Valencia-García, R. A collaborative filtering based recommender system for disease self-management. In *Technologies and Innovation* (Cham, 2017), R. Valencia-García, K. Lagos-Ortiz, G. Alcaraz-Mármol, J. Del Cioppo, N. Vera-Lucio, and M. Bucaram-Leverone, Eds., Springer International Publishing, pp. 60–71.

[82] Mimouni, N., Nazarenko, A., and Salotti, S. *A Conceptual Approach for Relational IR: Application to Legal Collections*. Springer International Publishing, Cham, 2015, pp. 303–318.

[83] Missaoui, R., Nourine, L., and Renaud, Y. An inference system for exhaustive generation of mixed and purely negative implications from purely positive ones. In *Proceedings of the 7th International Conference on Concept Lattices and Their Applications, Sevilla, Spain, October 19-21, 2010* (2010), pp. 271–282.

[84] Missaoui, R., Nourine, L., and Renaud, Y. Computing implications with negation from a formal context. *Fundam. Inf. 115*, 4 (Dec. 2012), 357–375.

[85] Mitchell, J. C. The implication problem for functional and inclusion dependencies. *Information and Control 56*, 3 (1983), 154 – 173.

[86] Mooney, R. J., and Roy, L. Content-based book recommending using learning for text categorization. In *Proceedings of the Fifth*

ACM Conference on Digital Libraries (New York, NY, USA, 2000), DL '00, ACM, pp. 195–204.

[87] Mora, A., Cordero, P., Enciso, M., Fortes, I., and Aguilera, G. Closure via functional dependence simplification. *International Journal of Computer Mathematics 89*, 4 (2012), 510–526.

[88] Morgan, C. G. An automated theorem prover for relational logic (abstract). In *Workshop Theorem Proving with Analytic Tableaux and Related Methods, Lautenbach. Universität Karlsruhe, Fakultät für Informatik, Institut für Logik, Komplexität und Deduktionssysteme, Interner Bericht 8/92, March 18-20, 1992* (1992), pp. 56–58.

[89] Nagler, T., and Czado, C. Evading the curse of dimensionality in nonparametric density estimation with simplified vine copulas. *Journal of Multivariate Analysis 151* (2016), 69 – 89.

[90] Nikolov, A., Ferrara, A., and Scharffe, F. Data linking for the semantic web. *Int. J. Semant. Web Inf. Syst. 7*, 3 (July 2011), 46–76.

[91] Outrata, J., and Vychodil, V. Fast algorithm for computing fixpoints of galois connections induced by object-attribute relational data. *Information Sciences 185*, 1 (2012), 114 – 127.

[92] Papadimitriou, A., Symeonidis, P., and Manolopoulos, Y. A generalized taxonomy of explanations styles for traditional and social recommender systems. *Data Min. Knowl. Discov. 24*, 3 (2012), 555–583.

[93] Paredaens, J., Bra, P., Gyssens, M., and Gucht, D. V., Eds. *The structure of the relational database model*. EATCS Monographs on Theoretical Computer Science, 1989.

[94] Pernelle, N., Saïs, F., and Symeonidou, D. An automatic key discovery approach for data linking. *Web Semantics: Science, Services and Agents on the WWW 23* (2013), 16–30.

[95] Poelmans, J., Dedene, G., Snoeck, M., and Viaene, S. An iterative requirements engineering framework based on formal concept analysis and c k theory. *Expert Systems with Applications 39*, 9 (2012), 8115 – 8135.

[96] Poelmans, J., Ignatov, D. I., Kuznetsov, S. O., and Dedene, G. Formal concept analysis in knowledge processing: A survey on applications. *Expert Systems with Applications 40*, 16 (2013), 6538 – 6560.

[97] Porcel, C., Tejeda-Lorente, A., Martínez, M. A., and Herrera-Viedma, E. A hybrid recommender system for the selective dissemination of research resources in a technology transfer office. *Inf. Sci. 184*, 1 (Feb. 2012), 1–19.

[98] Qu, K., Zhai, Y., Liang, J., and Chen, M. Study of decision implications based on formal concept analysis. *International Journal of General Systems 36*, 2 (2007), 147–156.

[99] Reda, A., Park, Y., Tiwari, M., Posse, C., and Shah, S. Metaphor: a system for related search recommendations. In *CIKM* (2012), X. wen Chen, G. Lebanon, H. Wang, and M. J. Zaki, Eds., ACM, pp. 664–673.

[100] Resnick, P., and Varian, H. R. Recommender systems. *Commun. ACM 40*, 3 (Mar. 1997), 56–58.

[101] Reusens, M., Lemahieu, W., Baesens, B., and Sels, L. Evaluating recommendation and search in the labor market. *Knowledge-Based Systems 152* (2018), 62 – 69.

[102] Ricci, F., Rokach, L., and Shapira, B. *Recommender Systems Handbook*, 2nd ed. Springer Publishing Company, 2015.

[103] Risch, V., and Schwind, C. Tableaux-based theorem proving and non-standard reasoning. In *Workshop Theorem Proving with Analytic Tableaux and Related Methods, Lautenbach. Universität Karlsruhe,*

Fakultät für Informatik, Institut für Logik, Komplexität und Deduktionssysteme, Interner Bericht 8/92, March 18-20, 1992 (1992), pp. 76–78.

[104] Sacarea, C., Sotropa, D., and Troanca, D. Symptoms investigation by means of formal concept analysis for enhancing medical diagnoses. In *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)* (Sept 2017), pp. 1–5.

[105] Saiedian, H., and Spencer, T. An efficient algorithm to compute the candidate keys of a relational database schema. *Comput. J. 39*, 2 (1996), 124–132.

[106] Sali, A. *Minimal Keys in Higher-Order Datamodels*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 242–251.

[107] Salimi, A., Ziaii, M., Amiri, A., Zadeh, M. H., Karimpouli, S., and Moradkhani, M. Using a feature subset selection method and support vector machine to address curse of dimensionality and redundancy in hyperion hyperspectral data classification. *The Egyptian Journal of Remote Sensing and Space Science 21*, 1 (2018), 27–36.

[108] Sallam, A., Bertino, E., Hussain, S. R., Landers, D., Lefler, R. M., and Steiner, D. Dbsafe: An anomaly detection system to protect databases from exfiltration attempts. *IEEE Systems Journal 11*, 2 (June 2017), 483–493.

[109] Senatore, S., and Pasi, G. Lattice navigation for collaborative filtering by means of (fuzzy) formal concept analysis. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing* (New York, NY, USA, 2013), SAC '13, ACM, pp. 920–926.

[110] Shah Khusro, Zafar Ali, I. U. Recommender systems: Issues, challenges, and research opportunities. In *Information Science and Applications (ICISA) 2016* (2016), vol. 376, IEEE, pp. 1179–1189.

[111] Sharma, R., and Ray, S. Explanations in recommender systems: An overview. *Int. J. Bus. Inf. Syst. 23*, 2 (Jan. 2016), 248–262.

[112] Simsion, G. C., and Witt, G. C. *Data modeling essentials*, 3rd ed. Amsterdam; Boston, 2005.

[113] Sismanis, Y., Brown, P., Haas, P. J., and Reinwald, B. Gordian: efficient and scalable discovery of composite keys. In *In Proc. International Conference on Very Large Data Bases (VLDB* (2006), pp. 691–702.

[114] Son, J., and Kim, S. B. Academic paper recommender system using multilevel simultaneous citation networks. *Decision Support Systems 105* (2018), 24 – 33.

[115] Son, L. H. Dealing with the new user cold-start problem in recommender systems: A comparative review. *Information Systems 58* (2016), 87 – 104.

[116] Sundaresan, N. Recommender systems at the long tail. In *Proceedings of the Fifth ACM Conference on Recommender Systems* (New York, NY, USA, 2011), RecSys '11, ACM, pp. 1–6.

[117] Tiroshi, A., Kuflik, T., Kay, J., and Kummerfeld, B. Recommender systems and the social web. In *UMAP Workshops* (2011), L. Ardissono and T. Kuflik, Eds., vol. 7138 of *Lecture Notes in Computer Science*, Springer, pp. 60–70.

[118] Trabelsi, W., Wilson, N., Bridge, D. G., and Ricci, F. Preference dominance reasoning for conversational recommender systems: a comparison between a comparative preferences and a sum of weights approach. *International Journal on Artificial Intelligence Tools 20*, 4 (2011), 591–616.

[119] Tu, X., Wang, Y., Zhang, M., and Wu, J. Using formal concept analysis to identify negative correlations in gene expression data.

IEEE-ACM Transactions on Computational Biology and Bioinformatics 13 (Jun 2015), 1 – 1.

[120] Ullman, J. D., and Widom, J. *A First Course in Database Systems*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[121] Vavilis, S., Egner, A., Petkovic, M., and Zannone, N. An anomaly analysis framework for database systems. *Computers and Security 53* (2015), 156 – 173.

[122] Verstrepen, K., Bhaduriy, K., Cule, B., and Goethals, B. Collaborative filtering for binary, positiveonly data. *SIGKDD Explor. Newsl. 19*, 1 (Sept. 2017), 1–21.

[123] Viegas, F., Rocha, L., Gonçalves, M., Mourão, F., Sá, G., Salles, T., Andrade, G., and Sandin, I. A genetic programming approach for feature selection in highly dimensional skewed data. *Neurocomputing 273* (2018), 554 – 569.

[124] Wastl, R. Linear derivations for keys of a database relation schema. *Journal of Universal Computer Science 4*, 12 (1998), 883–897.

[125] Wastl, R. On the number of keys of a relational database schema. *Journal of Universal Computer Science 4*, 5 (1998), 547–559.

[126] Wienke, S., Miller, J., Schulz, M., and Müller, M. S. Development effort estimation in hpc. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Piscataway, NJ, USA, 2016), SC '16, IEEE Press, pp. 10:1–10:12.

[127] Wille, R. *Formal Concept Analysis as Mathematical Theory of Concepts and Concept Hierarchies*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 1–33.

[128] Worland, P. B. An efficient algorithm for 3nf determination. *Information Sciences 167*, 1 (2004), 177 – 192.

[129] Yang, X., Wang, Z., Xue, J., and Zhou, Y. The reliability wall for exascale supercomputing. *IEEE Transactions on Computers 61*, 6 (June 2012), 767–779.

[130] Yang, Z., and Cai, Z. Detecting abnormal profiles in collaborative filtering recommender systems. *Journal of Intelligent Information Systems* (2016), 1–20.

[131] Yao, H., Hamilton, H. J., and Butz, C. J. Fd mine: Discovering functional dependencies in a database using equivalences. In *ICDM* (2002), IEEE Computer Society, pp. 729–732.

[132] Ying, Y. The personalized recommendation algorithm based on item semantic similarity. In *Communication Systems and Information Technology* (Berlin, Heidelberg, 2011), M. Ma, Ed., Springer Berlin Heidelberg, pp. 999–1004.

[133] Yu, C. T., and Johnson, D. T. On the complexity of finding the set of candidate keys for a given set of functional dependencies. *Inf. Process. Lett. 5*, 4 (1976), 100–101.

[134] Zamani, H., and Shakery, A. A language model-based framework for multi-publisher content-based recommender systems. *Information Retrieval Journal* (Online Feb 2018).

[135] Zhang, W., Du, Y. J., and Song, W. Recommender system with formal concept analysis. In *2015 International Conference on Information and Communications Technologies (ICT 2015)* (April 2015), pp. 1–6.

[136] Zhang, Y. Determining all candidate keys based on karnaugh map. *IEEE International Conference on Information Management, Innovation Management and Industrial Engineering 04* (2009), 226–229.

[137] Zhou, W., Wen, J., Gao, M., Liu, L., Cai, H., and Wang, X. *A Shilling Attack Detection Method Based on SVM and Target Item*

Analysis in Collaborative Filtering Recommender Systems. Springer International Publishing, Cham, 2015, pp. 751–763.

[138] Zobel, J. How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 1998), SIGIR '98, ACM, pp. 307–314.

[139] Zou, C., Zhang, D., Wan, J., Hassan, M. M., and Lloret, J. Using concept lattice for personalized recommendation system design. *IEEE Systems Journal 11*, 1 (March 2017), 305–314.

-¿Qué te parece desto, Sancho? –Dijo Don Quijote–
¿Hay encantos que valgan contra la verdadera valentía?
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

El Ingenioso Hidalgo Don Quijote de la Mancha

Miguel de Cervantes