

UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERA TÉCNICA EN INFORMÁTICA DE GESTIÓN

**ALGORITMOS PARA MANIPULACIÓN DE IMPLICACIONES EN
ANÁLISIS DE CONCEPTOS FORMALES**

Realizado por

ADORACIÓN M^a CALDERÓN RIVAS

Dirigido por

ÁNGEL MORA BONILLA

Departamento

MATEMÁTICA APLICADA

MÁLAGA, Diciembre 2015

UNIVERSIDAD DE MÁLAGA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
INGENIERA TÉCNICA EN INFORMÁTICA DE GESTIÓN

Reunido el tribunal examinador en el día de la fecha, constituido por:

Presidente/a

Dº/Dª. _____

Secretario/a

Dº/Dª. _____

Vocal

Dº/Dª. _____

para juzgar el proyecto Fin de Carrera titulado:

ALGORITMOS PARA MANIPULACIÓN DE IMPLICACIONES EN ANÁLISIS DE CONCEPTOS FORMALES

realizado por Dª.

Adoración Mª Calderón Rivas

tutorizado por Dº.

Ángel Mora Bonilla

y, en su caso, dirigido académicamente por

Dº/Dª. _____

ACORDÓ POR _____ OTORGAR LA CALIFICACIÓN

DE _____

Y PARA QUE CONSTE, SE EXTIENDE FIRMADA POR LOS COMPARCIENTES
DEL TRIBUNAL, LA PRESENTE DILIGENCIA.

Málaga a _____ de_____ del 2015

El presidente

El secretario

El vocal

Índice general

1. INTRODUCCIÓN	1
1.1. Historia	2
1.2. Contextos y conceptos	3
1.3. Implicaciones y reglas de asociación con FCA	4
1.4. Aplicaciones	4
2. ANÁLISIS DE CONCEPTOS FORMALES	7
2.1. ¿Qué es FCA?	8
2.2. Contexto formal	8
2.3. Operadores de derivación	9
2.4. Conceptos Formales	10
2.5. Retículo de conceptos de un contexto	12
2.6. Implicaciones de atributos	13
2.7. Estructuras matemáticas relacionadas con el FCA	16
2.7.1. Conexiones de Galois	16
2.7.2. Operadores de cierre	17
2.7.3. Extensiones, intensiones, retículo de conceptos	17
2.7.4. Definición concisa de conexiones de Galois	18
2.7.5. Conexiones de Galois, unión e intersección	18
2.7.6. Cada conexión de Galois es inducida por una relación binaria .	18
2.7.7. Teorema de representación para conexiones de Galois	19
2.7.8. Estructura jerárquica de retículo de conceptos	19
2.8. Teorema Principal de Retículo de Conceptos	20
2.8.1. Etiquetado de diagramas de retículos de conceptos	21
2.9. Clarificación y reducción de conceptos formales	22
2.9.1. Reducción de contextos formales por relaciones de vectores .	26
2.10. Algoritmos para el cálculo de retículos de conceptos	30
2.10.1. Algoritmo Next-Closure	31
2.10.2. Algoritmo UpperNeighbor	33
2.11. Contextos multivaluados y escalamiento conceptual	34

2.11.1. Escalamiento conceptual	36
3. ALGORITMOS DE CÁLCULO DE BASES	41
3.1. CLA	44
3.2. Direct Optimal Basis	46
3.3. Implementaciones	47
4. HERRAMIENTA PARA IMPLICACIONES	51
4.1. IS Bench	52
4.1.1. Workspaces	54
4.1.2. Registrar Benchmarks	55
4.1.3. Ejecutar algoritmos: Modos de ejecución	57
4.1.4. Ejecutar Benchmarks	58
4.1.5. Consulta de Resultados	61
4.2. Generador de sistemas implicacionales aleatorios	62
4.2.1. Generar un sistema implicacional	63
4.2.2. Generar n sistemas implicacionales	65
4.3. Ejemplo de uso	66
4.4. Implementación de Algoritmos	69
5. COMPARATIVA ENTRE ALGORITMOS	73
6. DISEÑO E IMPLEMENTACIÓN DE LA APLICACIÓN	75
6.1. Tecnologías y herramientas empleadas	76
6.2. Casos de uso	76
6.2.1. Inicio	78
6.2.2. Workspaces	79
6.2.3. Benchmarks	87
6.2.4. Resultados	103
6.3. Diagrama de componentes	105
6.4. Diagramas de paquetes	105
6.5. Diagramas de clases	107
6.5.1. Estereotipos	107
6.5.2. Dominio	108
6.5.3. Workspaces	128
6.5.4. Registrar Benchmarks	148
6.5.5. Ejecutar Benchmarks	172
6.5.6. Generador Implicaciones	199
6.5.7. Resultados	221
6.5.8. API Algoritmos	243

6.6. Diagramas de secuencia 258

7. CONCLUSIONES 265

Capítulo 1

INTRODUCCIÓN

1.1. Historia

El Análisis de Conceptos Formales (FCA) constituye una herramienta formal para el análisis de datos que permite la extracción de conocimiento a partir de un conjunto de objetos y de las propiedades que cumplen dichos objetos.

La teoría en su forma actual se remonta al grupo de investigación dirigido por Rudolf Darmstadt Wille, Bernhard Ganter y Peter Burmeister, donde se originó el análisis de conceptos formales en la década de 1980. La base matemática, sin embargo, ya se había creado por Garrett Birkhoff en la década de los 30 como parte de la teoría general de retículos. Antes del trabajo del grupo de Darmstadt, ya existían enfoques similares de varios grupos franceses, y sus fundamentos filosóficos apuntan principalmente a Charles S. Peirce y al pedagogo Hartmut von Hentig.

En su artículo “Reestructuración de la Teoría de Retículos” (de 1982, con el que se inicia el FCA como disciplina matemática) Rudolf Wille parte de un descontento con la teoría de retículos en particular y con las matemáticas puras en general.

La producción de resultados teóricos - a menudo alcanzados por medio de elaboradas “gimnasias mentales” - eran impresionantes, pero las conexiones entre dominios vecinos, o incluso entre partes de una misma teoría se estaban debilitando. La reestructuración de la teoría de retículos es un intento de revitalizar las conexiones entre dominios mediante la reinterpretación de la teoría de la manera más concreta posible con el fin de propiciar una mejor comunicación entre los teóricos de retículos y los usuarios potenciales de dicha teoría. Este objetivo se remonta a Hartmut von Hentig, quien en 1972 pidió una reestructuración de las ciencias con el fin de conseguir una mejor enseñanza y de hacer a la ciencia más asequible, tanto con el fin de conocerla mejor como para poder criticarla de una forma más cercana (es decir, sin necesidad de un conocimiento especializado).

El FCA corrige el punto de partida de la teoría reticular en el desarrollo de la lógica formal en el siglo XIX, donde un concepto, como predicado unario, se había reducido a su extensión. El objetivo fue trabajar con una visión de los conceptos menos abstracta, haciendo uso también de la intención, una orientación que provenía de la lingüística y de la lógica conceptual clásica.

FCA tiene como objetivo aclarar los conceptos siguiendo la máxima de Charles S. Peirce de desplegar las propiedades observables y elementales de los objetos. En su filosofía tardía, Peirce supone que el pensamiento lógico tiene como objetivo percibir la realidad por medio de la triada: concepto, juicio y conclusión. En este sentido, Wille dice:

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.

1.2. Contextos y conceptos

El FCA tiene como entrada una tabla con una relación entre un conjunto de objetos y un conjunto de propiedades (atributos).

Esta entrada es lo que se denomina **contexto formal** y la relación que se define en dicha tabla determina qué objetos cumplen qué propiedades.

Un concepto formal se define como un par formado por un conjunto de objetos (**extensión**) y un conjunto de atributos (**intensión**) de forma que la extensión está formada por todos los objetos que comparten los atributos dados y la intensión por los atributos compartidos por los objetos dados.

Por lo que, un **contexto formal** se puede definir como una tripleta $K=(X, Y, I)$, donde X es un conjunto de objetos, Y es un conjunto de atributos e I una relación binaria, $I \subseteq X \times Y$, que muestra qué objeto posee qué atributo. Formalmente, se puede considerar como un grafo bipartito que refleje las relaciones entre los conjuntos X y Y , o también como una tabla con los objetos ocupando las filas y los atributos en las columnas, y de forma que un valor booleano en la celda (x, y) significa que el objeto x tiene el atributo y .

	necesita agua para vivir	vive en el agua	vive en la tierra	necesita clorofila	dicotiledónea	monocotiledónea	puede moverse	tiene extremidades	lactantes
sanguijuela	×	×					×		
brema	×	×					×	×	
rana	×	×	×				×	×	
perro	×		×				×	×	
maleza acuática	×	×		×		×			
caña	×	×	×	×		×		×	
judía	×		×	×	×				
maíz	×		×	×		×			

Contexto Formal

1.3. Implicaciones y reglas de asociación con FCA

Una vez formalizados matemáticamente los conceptos, puede resultar relativamente sencillo trasladar las relaciones lógicas que encontramos entre atributos... aunque debemos tener en cuenta que las relaciones lógicas que se obtengan son aquellas que vengan confirmadas por nuestro contexto, que podría representar un conocimiento concreto en un instante de tiempo, pero que podría variar al añadir un mayor número de objetos o atributos. Por ejemplo, que en el ejemplo siguiente podamos decir que “ser animal de jungla” implica “ser mamífero” se debe a que todos los objetos que verifican la propiedad “ser animal de jungla” verifican “ser mamífero”, pero en el momento que introduzcamos un objeto nuevo que sea de la jungla (por ejemplo, un manglar) y no sea mamífero, esa implicación deja de ser cierta. Por supuesto, podemos trabajar con varios atributos simultáneamente.

Consecuentemente, en FCA podemos formalizarlo de la siguiente forma: dados A y B subconjuntos de atributos, diremos que se tiene la implicación $A \rightarrow B$ si se verifica $A' \subseteq B'$, es decir, todos los objetos que tienen cada atributo de A también tienen cada atributo de B (observa que es coherente con la implicación intuitiva que dimos en el apartado anterior).

Con esta definición, las implicaciones obedecen las reglas de Armstrong (reflexiva, aumentativa y transitiva) comunes en las dependencias funcionales que se dan entre los atributos de una base de datos:

Reflexividad: $B \subseteq A \quad A \rightarrow B$

Aumento: $A \rightarrow B \quad A \cup C \rightarrow B \cup C$

Transitividad: $A \rightarrow B, B \rightarrow C \quad A \rightarrow C$

A partir de la definición de implicación y de las propiedades básicas que verifica, podemos definir un cálculo lógico que nos permitirá realizar sistemas de deducción completos sobre el contexto actual. En cierta forma, hemos pasado de tener un conocimiento por ejemplos a disponer de un conocimiento abstracto que introduce sistemas de razonamiento más elaborados en nuestro mundo, partiendo únicamente de las observaciones concretas que hemos realizado, es decir, hemos aprendido reglas generales a partir de ejemplos.

1.4. Aplicaciones

Desde su introducción ha sido aplicado en campos tan variados como la minería de datos, minería de textos, gestión del conocimiento, web semántica, desarrollo de software, biología, etc.

La doctora Karelle Bertet, investigadora de la Universidad de La Rochelle con la que colabora el director del proyecto, ha desarrollado una librería java, java-lattices, para la generación, representación y manipulación de los Conceptos Formales, etc.

Dicha librería implementa la generación de retículos. Se pueden generar:

- Un retículo dados los nodos y sus relaciones.
- Álgebras booleanas de 2^n elementos.
- Retículos de permutaciones.
- Retículos aleatorios de n nodos.
- Retículos de conceptos a partir de un contexto o de un conjunto de implicaciones.
- Cálculo de implicaciones y de bases de implicaciones.
- Etc.

Capítulo 2

ANÁLISIS DE CONCEPTOS FORMALES

2.1. ¿Qué es FCA?

El Análisis de Conceptos Formales (FCA) es un método de análisis de datos con creciente popularidad en distintos ámbitos como puede ser la minería de datos, preprocesamiento de datos o descubrimiento del conocimiento.

FCA analiza los datos que describen relaciones entre un conjunto de objetos y un conjunto de atributos y genera dos tipos de salida a partir de dichos datos.

El primer tipo es un **retículo de conceptos**. Un retículo de conceptos es un conjunto de conceptos formales ordenados jerárquicamente por una relación subconcepto-superconcepto.

Los conceptos formales son agrupaciones que representan objetos, cosas, nociones como “organismos que viven en el agua”, “números divisibles por 3 y 4”, etc. Formalmente, se pueden definir como pares $\langle A, B \rangle$ donde $A \subseteq X$ es un conjunto de objetos y $B \subseteq Y$ es un conjunto de atributos, tal que todos los elementos de A son todos los objetos que tienen los atributos de B y los elementos de B son los atributos comunes a todos los objetos de A .

El segundo tipo de salida es un **conjunto de implicaciones de atributos**.

Las implicaciones de atributos describen una dependencia concreta válida en los datos de entrada, como pueden ser “todos los números divisibles por 3 y 4 son divisibles por 6”, “todo encuestado mayor de 60 está jubilado”, etc.

La característica que diferencia a este método de análisis de datos de otros es la inherente integración entre tres componentes del procesamiento de datos y del conocimiento:

- el descubrimiento y razonamiento con conceptos en los datos,
- el descubrimiento y razonamiento de dependencias en los datos,
- y la visualización de los datos, conceptos y dependencias.

2.2. Contexto formal

Un contexto formal es la entrada de datos de FCA y se puede definir como una tripleta $\langle X, Y, I \rangle$, donde X es un **conjunto de objetos**, Y es un **conjunto de atributos** e I una **relación binaria**, $I \subseteq X \times Y$, que muestra qué objeto posee qué atributo. Como se ha dicho en la introducción, formalmente se puede considerar como un grafo bipartito que refleje las relaciones entre los conjuntos X y Y , o también como una tabla con los objetos ocupando las filas y los atributos en las columnas, y de forma que un valor booleano en la celda (x,y) significa que el objeto x tiene el atributo y .

Ejemplo: En la 2.1, se muestra un ejemplo de contexto formal representado como tabla.

I	y_1	y_2	y_3	y_4
x_1	×	×	×	×
x_2	×		×	×
x_3		×	×	×
x_4		×	×	×
x_5	×			

Figura 2.1: Representación en tabla de un Contexto Formal

En la tabla \times indica que un objeto tiene un determinado atributo así como un espacio en blanco indica que el objeto no tiene el atributo. Por ejemplo, si los objetos son coches y los atributos son características de los coches tal que y_1 es el atributo "tener ABS", los coches x_1, x_2 y x_5 tendrían ABS pero x_3 y x_4 no.

Una tabla con atributos lógicos puede ser representado por una tripleta $\langle X, Y, I \rangle$ donde I es una relación binaria entre X e Y .

2.3. Operadores de derivación

Con el fin de poder trabajar más cómodamente con la relación I , definimos los operadores de derivación $\uparrow: 2^X \rightarrow 2^Y$ y $\downarrow: 2^Y \rightarrow 2^X$ como:

$$A^\uparrow = \{y \in Y : \forall x \in A \langle x, y \rangle \in I\} \text{ donde } A \subseteq X$$

$$B^\downarrow = \{x \in X : \forall y \in B \langle x, y \rangle \in I\} \text{ donde } B \subseteq Y$$

El operador $\uparrow\downarrow$ verifica algunas propiedades interesantes que serán fundamentales para poder desarrollar la teoría formal (matemáticamente, por verificar las siguientes propiedades, decimos que es un operador de cierre):

- idempotencia: $A^{\uparrow\downarrow\uparrow\downarrow} = A^{\uparrow\downarrow}$,
- monotonía: $A_1 \subseteq A_2 \rightarrow A_1^{\uparrow\downarrow} \subseteq A_2^{\uparrow\downarrow}$,
- extensividad: $A \subseteq A^{\uparrow\downarrow}$

Obsérvese que, en general, no se verifica que $A^{\uparrow\downarrow} = A$. Cuando un conjunto de objetos, $A \subseteq X$ verifica que $A^{\uparrow\downarrow} = A$ se llama cerrado. Una definición similar se obtiene para hablar de conjuntos de atributos cerrados, es decir, subconjuntos del conjunto Y .

Ejemplo: Para la tabla de la Figura 2.1 tenemos:

$$\begin{aligned}
 \{x_2\}^\uparrow &= \{y_1, y_3, y_4\}, \{x_2, x_3\}^\uparrow = \{y_3, y_4\}, \\
 \{x_1, x_4, x_5\}^\uparrow &= \emptyset, \\
 X^\uparrow &= \emptyset, \emptyset^\uparrow = Y, \\
 \{y_1\}^\downarrow &= \{x_1, x_2, x_5\}, \{y_1, y_2\}^\downarrow = \{x_1\}, \\
 \{y_2, y_3\}^\downarrow &= \{x_1, x_3, x_4\}, \{y_2, y_3, y_4\}^\downarrow = \{x_1, x_3, x_4\}, \\
 \emptyset^\downarrow &= X, Y^\downarrow = \{x_1\}.
 \end{aligned}$$

2.4. Conceptos Formales

El Concepto Formal es la noción básica del FCA. La definición de Concepto Formal se puede hacer desde varios enfoques:

- Psicología

Murphy G. L.: *The Big Book of Concepts*. MIT Press, 2004.
 Margolis E., Laurence S.: *Concepts: Core Readings*. MIT Press, 1999.
- Lógica

Tichy P.: *The Foundations of Frege's Logic*. W. De Gryuter, 1988.
 Materna P.: *Conceptual Systems*. Logos Verlag, Berlin, 2004.
- Inteligencia Artificial (aprendizaje de conceptos)

Michalski, R. S., Bratko, I. and Kubat, M. (Eds.), *Machine Learning and Data Mining: Methods and Applications*, London, Wiley, 1998.
- Grafos conceptuales

Sowa J. F.: *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Course Technology, 1999.
- Modelado conceptual, paradigma orientado a objetos, ...
- Tradicional / Lógica de Port-Royal

Arnauld A., Nicole P.: *La logique ou l'art de penser*, 1662 (*Logic Or The Art Of Thinking*, CUP, 2003).

La noción de concepto usado en FCA está inspirado en la Lógica de Port-Royal (lógica tradicional) y se define como un par formado por un conjunto de objetos (**extensión**) y un conjunto de atributos (**intensión**) de forma que la extensión está formada por todos los objetos que comparten los atributos dados y la intensión por los atributos

compartidos por los objetos dados.

Ejemplo:

- concepto: COCHE
- extensión: conjunto de todos los coches (Mercedes, Nissan, Toyota,...)
- intención: tiene motor, tiene asientos, tiene ruedas, ...

A partir de los operadores de derivación definidos en el apartado anterior se puede decir que un par $\langle A, B \rangle$ se llama un concepto formal de un contexto si verifica:

- $A \subseteq X, B \subseteq Y,$
- $A^\uparrow = B, B^\uparrow = A.$

Aunque puede parecer una definición un poco arbitraria, intuitivamente un par, $\langle A, B \rangle$, es un concepto en el contexto si:

- cada objeto de A tiene todos los atributos de B ,
- para cada objeto en X que no está en A , existe un atributo en B que el objeto no tiene,
- para cada atributo en Y que no está en B , hay un objeto en A que no tiene ese atributo.

Luego, en cierta forma, conseguimos introducir en la definición formal de concepto las dos partes que filosóficamente considerábamos esenciales: por una parte, el conjunto de objetos con propiedades comunes, y por otra el conjunto de atributos que caracterizan a dichos objetos. Únicamente aquellos pares de conjuntos que tienen una conexión perfectamente cerrada establecen un concepto por sí mismos. Allí donde no hay atributos ausentes ni contraejemplos entre sus objetos. En esta situación, los conjuntos A y B son cerrados y se llaman, respectivamente, la extensión y la intención del concepto. Para un conjunto de objetos, A , el conjunto de sus atributos comunes, A^\uparrow , describe de alguna forma la similitud de los objetos de A , mientras que el conjunto $A^{\uparrow\downarrow}$ es la agrupación de objetos que tienen como atributos comunes a A^\uparrow (en particular, estarán todos los objetos de A , es decir, $A \subseteq A^{\uparrow\downarrow}$).

Por tanto, un concepto, en la representación matricial, se puede reconocer por medio de una submatriz maximal (no necesariamente formada por celdas contiguas) de tal manera que todas las celdas de la submatriz son verdaderas.

<i>I</i>	<i>y</i> ₁	<i>y</i> ₂	<i>y</i> ₃	<i>y</i> ₄
<i>x</i> ₁	✗	✗	✗	✗
<i>x</i> ₂	✗		✗	✗
<i>x</i> ₃		✗	✗	✗
<i>x</i> ₄		✗	✗	✗
<i>x</i> ₅	✗			

Figura 2.2: Concepto Formal

En la Figura 2.2 la zona sombreada representa el concepto formal

$\langle A_1, B_1 \rangle = \langle \{x_1, x_2, x_3, x_4\}, \{y_3, y_4\} \rangle$ porque $\{x_1, x_2, x_3, x_4\}^\uparrow = \{y_3, y_4\}$ y $\{y_3, y_4\}^\downarrow = \{x_1, x_2, x_3, x_4\}$.

En la representación como grafo bipartito se reconocerá como subgrafo bipartito completo (es decir, aquel que tiene todas las aristas posibles). Una definición más formal sería que $\langle A, B \rangle$ es un concepto formal si y sólo si $\langle A, B \rangle$ es un punto fijo de $\langle \uparrow, \downarrow \rangle$.

2.5. Retículo de conceptos de un contexto

En el conjunto de todos los conceptos puede establecerse una relación de orden relacionada con las nociones de subconcepto y superconcepto. Esta relación de orden está basada en la relación de inclusión de objetos y atributos. Su definición es la que sigue:

Para los conceptos formales $\langle A_1, B_1 \rangle, \langle A_2, B_2 \rangle \in \langle X, Y, I \rangle$, se cumple que $\langle A_1, B_1 \rangle \leq \langle A_2, B_2 \rangle \Leftrightarrow A_1 \subseteq A_2 (\Leftrightarrow B_2 \subseteq B_1)$

- \leq representa la ordenación subconcepto-superconcepto.
- $\langle A_1, B_1 \rangle \leq \langle A_2, B_2 \rangle$ significa que $\langle A_1, B_1 \rangle$ es más específico que $\langle A_2, B_2 \rangle$ ($\langle A_2, B_2 \rangle$ es más general).

Por ejemplo, dados los conceptos COCHE y VEHICULO tendríamos la relación $COCHE \leq VEHICULO$ (el concepto COCHE es más específico que el concepto VEHICULO).

Cada par de conceptos en este orden parcial tiene una única máxima cota inferior, que es el concepto generado por $A_1 \cap A_2$. Simétricamente, cada par de conceptos en este orden parcial tiene una única mínima cota superior, que es el concepto generado por los atributos $B_1 \cap B_2$.

Estas operaciones que calculan el máximo y mínimo de dos conceptos satisfacen los axiomas que definen un retículo, y es fácil probar que cualquier retículo finito puede ser generado como el retículo de conceptos de algún contexto (por ejemplo, si L es el

retículo, se crea un contexto en el que $X=Y=L$ y la relación $\langle x, y \rangle \in I \Leftrightarrow x \leq y$ en el retículo).

Ejemplo: Consideremos el siguiente contexto formal:

	a	b	c	d	e	f	g	h	i
sanguijuela	1	x	x				x		
brema	2	x	x				x	x	
rana	3	x	x	x			x	x	
perro	4	x		x			x	x	x
maleza acuática	5	x	x		x		x		
caña	6	x	x	x	x		x		
haba	7	x		x	x	x			
maíz	8	x		x	x		x		

Figura 2.3: ContextoFormal

a: necesita el agua para vivir, b: vive en el agua, c: vive en la tierra, d: necesita clorofila para producir comida, e: dos hojas de la semilla, f: hoja de una semilla, g: puede moverse, h: tiene extremidades, i: amamanta a sus crías

Del anterior contexto formal $\langle X, Y, I \rangle$ se pueden deducir los siguientes conceptos formales:

$$\begin{aligned}
 C_0 &= \langle \{1, 2, 3, 4, 5, 6, 7, 8\}, \{a\} \rangle, C_1 = \langle \{1, 2, 3, 4\}, \{a, g\} \rangle, C_2 = \langle \{2, 3, 4\}, \{a, g, h\} \rangle, \\
 C_3 &= \langle \{5, 6, 7, 8\}, \{a, d\} \rangle, C_4 = \langle \{5, 6, 8\}, \{a, d, f\} \rangle, C_5 = \langle \{3, 4, 6, 7, 8\}, \{a, c\} \rangle, \\
 C_6 &= \langle \{3, 4\}, \{a, c, g, h\} \rangle, C_7 = \langle \{4\}, \{a, c, g, h, i\} \rangle, C_8 = \langle \{6, 7, 8\}, \{a, c, d\} \rangle, \\
 C_9 &= \langle \{6, 8\}, \{a, c, d, f\} \rangle, C_{10} = \langle \{7\}, \{a, c, d, e\} \rangle, C_{11} = \langle \{1, 2, 3, 5, 6\}, \{a, b\} \rangle, \\
 C_{12} &= \langle \{1, 2, 3\}, \{a, b, g\} \rangle, C_{13} = \langle \{2, 3\}, \{a, b, g, h\} \rangle, C_{14} = \langle \{5, 6\}, \{a, b, d, f\} \rangle, \\
 C_{15} &= \langle \{3, 6\}, \{a, b, c\} \rangle, C_{16} = \langle \{3\}, \{a, b, c, g, h\} \rangle, C_{17} = \langle \{6\}, \{a, b, c, d, f\} \rangle, \\
 C_{18} &= \langle \{\}, \{a, b, c, d, e, f, g, h, i\} \rangle
 \end{aligned}$$

El correspondiente retículo de conceptos $B(X, Y, I)$ se muestra en la Figura 2.4:

2.6. Implicaciones de atributos

Las implicaciones de atributos representan dependencias entre ellos tales como:

- todos los números divisibles por 2 y por 3 son divisibles por 6
- todos los pacientes con el síntoma s_2 y el síntoma s_5 tienen también el síntoma s_1 y el s_3 .

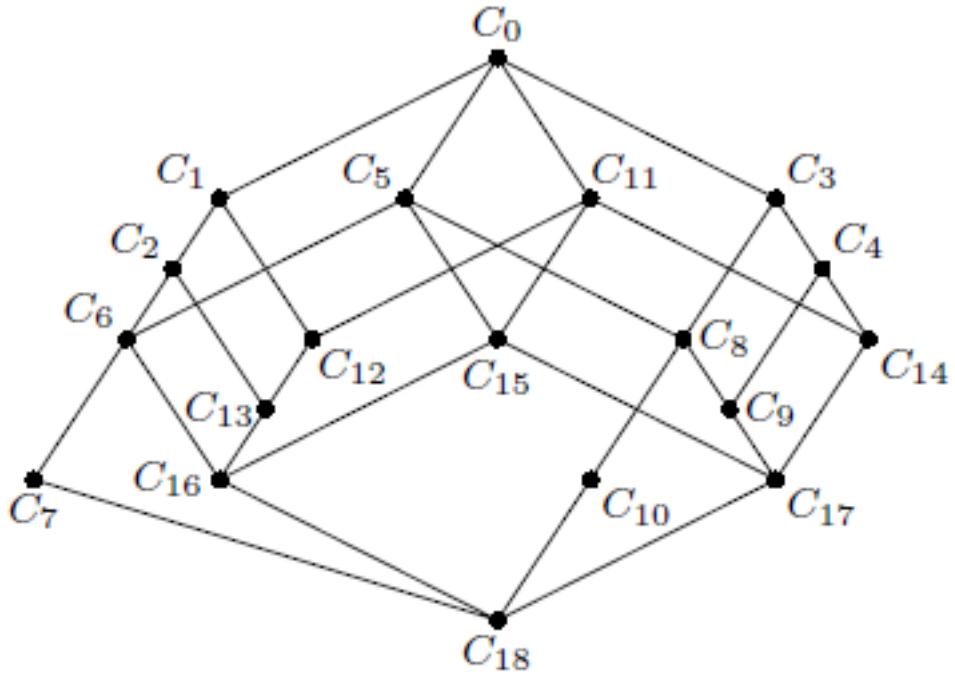


Figura 2.4: Retículo de conceptos $B(X, Y, I)$

Siendo Y un conjunto de atributos no vacío, una implicación de atributos en Y es una expresión

$$A \Rightarrow B \text{ donde } A \subseteq Y \text{ y } B \subseteq Y$$

Ejemplos:

- Dado $Y = \{y_1, y_2, y_3, y_4\}$, $\{y_2, y_3\} \Rightarrow \{y_1, y_4\}, \{y_2, y_3\} \Rightarrow \{y_1, y_2, y_3\}, \emptyset \Rightarrow \{y_1, y_2\}, \{y_2, y_4\} \Rightarrow \emptyset$ son implicaciones de atributos en Y .
- Dado $Y = \text{ver la tele, comer comida basura, correr regularmente, presión arterial normal, presión arterial alta}$, entonces $\{\text{ver la tele}\} \Rightarrow \{\text{presión arterial alta}\}$, $\{\text{correr regularmente} \Rightarrow \text{presión arterial normal}\}$ son implicaciones de atributos en Y .

La estructura con la cual evaluamos las implicaciones de atributos son las filas de las tablas de los contextos formales, que pueden considerarse como el conjunto de atributos de cada objeto.

Por lo que, una implicación de atributos $A \Rightarrow B$ en el conjunto de atributos Y es válida en un conjunto si y sólo si

$$A \subseteq M \Rightarrow B \subseteq M$$

- Escribimos

$$|A \Rightarrow B|_{\mathcal{M}} \left\{ \begin{array}{l} 1 \text{ si } A \Rightarrow B \text{ es verdadero en } \mathcal{M}, \\ 0 \text{ si } A \Rightarrow B \text{ no es verdadero en } \mathcal{M}. \end{array} \right.$$

- Sea M un conjunto de atributos de algún objeto x . $|A \Rightarrow B|_M$ quiere decir que "si x tiene todos los atributos de A , entonces x tiene todos los atributos de B ". Por lo que, $C \subseteq M$ es lo mismo que decir "si x tiene todos los atributos de C " .

Ejemplo: Dado $Y = \{y_1, y_2, y_3, y_4\}$

$A \Rightarrow B$	M	$ A \Rightarrow B _M$	why
$\{y_2, y_3\} \Rightarrow \{y_1\}$	$\{y_2\}$	1	$A \not\subseteq M$
$\{y_2, y_3\} \Rightarrow \{y_1\}$	$\{y_1, y_2\}$	1	$A \not\subseteq M$
$\{y_2, y_3\} \Rightarrow \{y_1\}$	$\{y_1, y_2, y_3\}$	1	$A \subseteq M \text{ and } B \subseteq M$
$\{y_2, y_3\} \Rightarrow \{y_1\}$	$\{y_2, y_3, y_4\}$	0	$A \subseteq M \text{ but } B \not\subseteq M$
$\{y_2, y_3\} \Rightarrow \{y_1\}$	\emptyset	1	$A \not\subseteq \emptyset$
$\emptyset \Rightarrow \{y_1\}$	$\{y_1, y_4\}$	1	$\emptyset \subseteq M \text{ and } B \subseteq M$
$\emptyset \Rightarrow \{y_1\}$	$\{y_3, y_4\}$	0	$\emptyset \subseteq M \text{ but } B \not\subseteq M$
$\{y_2, y_3\} \Rightarrow \emptyset$	any M	1	$\emptyset \subseteq M$

Figura 2.5: Implicaciones de atributos

Dado $\mathcal{M} \subseteq 2^Y$, una implicación $A \Rightarrow B$ en Y es válida en \mathcal{M} si es verdadero en cada $M \in \mathcal{M}$.

- De nuevo,

$$|A \Rightarrow B|_{\mathcal{M}} \left\{ \begin{array}{l} 1 \text{ si } A \Rightarrow B \text{ es verdadero en } \mathcal{M}, \\ 0 \text{ si } A \Rightarrow B \text{ no es verdadero en } \mathcal{M}. \end{array} \right.$$

por lo tanto, $|A \Rightarrow B|_{\mathcal{M}} = \min_{M \in \mathcal{M}} |A \Rightarrow B|_M$

Así llegamos a la definición de la validez de una implicación de atributos en un contexto formal.

Una implicación $A \Rightarrow B$ en Y es verdadera en un contexto formal $\langle X, Y, I \rangle$ si y sólo si $A \Rightarrow B$ es verdadera en

$$|A \Rightarrow B|_{\langle X, Y, I \rangle} = 1 \text{ si } A \Rightarrow B \text{ es verdadero en } \langle X, Y, I \rangle$$

- $\{x\}^\uparrow$ es el conjunto de atributos de x . Por lo que, $\mathcal{M} = \{\{x\}^\uparrow : x \in X\}$ es la colección cuyos elementos son justo los conjuntos de atributos de los objetos de $\langle X, Y, I \rangle$, es decir las filas de la tabla del contexto formal. Así que, $|A \Rightarrow B|_{\langle X, Y, I \rangle} = 1$ si y sólo si $A \Rightarrow B$ es verdadero en cada fila de $\langle X, Y, I \rangle$ si y sólo si para cada fila de $\langle X, Y, I \rangle$: si x tiene todos los atributos de A entonces x tiene todos los atributos de B .

Con esta definición, las implicaciones obedecen las reglas de Armstrong (reflexiva, aumentativa y transitiva) comunes en las dependencias funcionales que se dan entre los atributos de una base de datos:

$$\frac{B \subseteq A}{A \Rightarrow B}, \frac{A \Rightarrow B}{A \cup C \Rightarrow B \cup C^{\downarrow}}, \frac{D \Rightarrow B, B \Rightarrow C}{A \Rightarrow C}$$

A partir de la definición de implicación y de las propiedades básicas que verifica, podemos definir un cálculo lógico que nos permitiría realizar sistemas de deducción completos sobre el contexto actual. En cierta forma, hemos pasado de tener un conocimiento por ejemplos a disponer de un conocimiento abstracto que introduce sistemas de razonamiento más elaborados en nuestro mundo, partiendo únicamente de las observaciones concretas que hemos realizado, es decir, hemos aprendido reglas generales a partir de ejemplos.

2.7. Estructuras matemáticas relacionadas con el FCA

En esta sección se describen las estructuras matemáticas en las que se basa el FCA y sus propiedades.

2.7.1. Conexiones de Galois

Conexión de Galois: Una conexión de Galois entre dos conjuntos X e Y es un par $\langle f, g \rangle$ de $f : 2^X \rightarrow 2^Y$ y $g : 2^Y \rightarrow 2^X$ cumpliendo para $A, A_1, A_2 \subseteq X, B, B_1, B_2 \subseteq Y$:

$$A_1 \subseteq A_2 \Rightarrow f(A_2) \subseteq f(A_1), \quad (2.1)$$

$$B_1 \subseteq B_2 \Rightarrow g(B_2) \subseteq g(B_1), \quad (2.2)$$

$$A \subseteq g(f(A)), \quad (2.3)$$

$$B \subseteq f(g(B)). \quad (2.4)$$

Puntos fijos de conexiones de Galois: Para una conexión de Galois $\langle f, g \rangle$ entre dos conjuntos X e Y , al conjunto

$$fix(\langle f, g \rangle) = \{ \langle A, B \rangle \in 2^X \times 2^Y | f(A) = B, g(B) = A \}$$

es llamado conjunto de puntos fijos de $\langle f, g \rangle$

Teorema 2.7.1 (operadores de derivación para conexiones de Galois). *Para un contexto formal $\langle X, Y, I \rangle$, el par de operadores $\langle \uparrow' \downarrow' \rangle$ inducidos por $\langle X, Y, I \rangle$ es una conexión*

de Galois entre X e Y .

Lema 2.7.2. (*encadenamiento de conexiones de Galois*) Para una conexión de Galois $\langle f, g \rangle$ entre dos conjuntos X e Y se cumple que $f(A) = f(g(f(A)))$ y $g(B) = g(f(g(B)))$ para cualquier $A \subseteq X$ y $B \subseteq Y$.

2.7.2. Operadores de cierre

Operador de cierre: Un operador de cierre en el conjunto X es un mapeo $C : 2^X \rightarrow 2^Y$ que cumple para cada $A, A_1, A_2 \subseteq X$

$$A \subseteq C(A), \tag{2.5}$$

$$A_1 \subseteq A_2 \Rightarrow C(A_1) \subseteq C(A_2), \tag{2.6}$$

$$C(A) \subseteq C(C(A)). \tag{2.7}$$

Puntos fijos de operadores de cierre: Para un operador de cierre $C : 2^X \rightarrow 2^Y$, al conjunto

$$\text{fix}(C) = \{A \subseteq X | C(A) = A\}$$

es llamado punto fijo de C .

Teorema 2.7.3 (desde conexiones de Galois a operadores de cierre). *Si $\langle f, g \rangle$ es una conexión de Galois entre X e Y entonces $C_X = f \circ g$ es un operador de cierre en X y $C_Y = g \circ f$ es un operador de cierre en Y .*

Teorema 2.7.4. (*extensiones e intensiones*)

$$\begin{aligned} \text{Ext}(X, Y, I) &= \{B^\downarrow | B \subseteq Y\}, \\ \text{Int}(X, Y, I) &= \{A^\uparrow | A \subseteq X\} \end{aligned}$$

Teorema 2.7.5. *La menor extensión que contiene a $A \subseteq X$ es $A^{\uparrow\downarrow}$. La menor intención que contiene a $B \subseteq Y$ es $B^{\downarrow\uparrow}$.*

2.7.3. Extensiones, intensiones, retículo de conceptos

Teorema 2.7.6. *Para cualquier contexto formal $\langle X, Y, I \rangle$*

$$\begin{aligned}
Ext(X, Y, I) &= fix(\uparrow\downarrow), \\
Int(X, Y, I) &= fix(\downarrow\uparrow), \\
\mathcal{B}(X, Y, I) &= \{\langle A, A^\uparrow \rangle \mid A \in Ext(X, Y, I)\}, \\
\mathcal{B}(X, Y, I) &= \{\langle B^\downarrow, A \rangle \mid B \in Int(X, Y, I)\}.
\end{aligned}$$

Observación

El teorema anterior dice: A fin de obtener $\mathcal{B}(X, Y, I)$ podemos:

1. procesar $Ext(X, Y, I)$,
2. para cada $A \in Ext(X, Y, I)$ generar $\langle A, A^\uparrow \rangle$.

2.7.4. Definición concisa de conexiones de Galois

Hay una condición simple la cual es equivalente a las condiciones (1) - (4) de la definición de conexión de Galois.

Teorema 2.7.7. $\langle f, g \rangle$ es una conexión de Galois entre X e Y si y sólo si para todo $A \subseteq X$ y $B \subseteq Y$:

$$A \subseteq g(B) \text{ sii } B \subseteq f(A) \quad (2.8)$$

2.7.5. Conexiones de Galois, unión e intersección

El siguiente teorema describe el comportamiento básico de las conexiones de Galois respecto a la unión y a la intersección.

Teorema 2.7.8. $\langle f, g \rangle$ es una conexión de Galois entre X e Y cuando $A_j \subseteq X, j \in J$ y $B_j \subseteq Y, j \in J$ tenemos que

$$f\left(\bigcup_{j \in J} A_j\right) = \bigcap_{j \in J} f(A_j), \quad (2.9)$$

$$g\left(\bigcup_{j \in J} B_j\right) = \bigcap_{j \in J} g(B_j), \quad (2.10)$$

2.7.6. Cada conexión de Galois es inducida por una relación binaria

No sólo todos los pares de operadores de derivación forman un Galois, todos las conexiones de Galois es un operador de derivación de un contexto formal particular.

Teorema 2.7.9. Sea $\langle f, g \rangle$ una conexión de Galois entre X e Y . Considerese un contexto formal $\langle X, Y, I \rangle$ tal que I está definido por

$$\langle x, y \rangle \in I \text{ si y sólo si } y \in f(\{x\}) \text{ o, equivalentemente, si } x \in g(\{y\}), \quad (2.11)$$

para cada $x \in X$ e $y \in Y$. Entonces $\langle \uparrow', \downarrow' \rangle = \langle f, g \rangle$, esto es, los operadores de derivación $\langle \uparrow', \downarrow' \rangle$ inducido por $\langle X, Y, I \rangle$ coincide con $\langle f, g \rangle$.

Observaciones

- La relación I inducida a partir de $\langle f, g \rangle$ por (11) se indicará por $I_{\langle f, g \rangle}$.
- De ahí que, hemos establecido dos mapeos:
 $I \mapsto \langle \uparrow', \downarrow' \rangle$ asigna una conexión de Galois a una relación binaria I . $\langle \uparrow, \downarrow \rangle \mapsto I_{\langle \uparrow, \downarrow \rangle}$ asigna una relación binaria a una conexión de Galois.

2.7.7. Teorema de representación para conexiones de Galois

Teorema 2.7.10. (*de representación*)

$I \mapsto \langle \uparrow', \downarrow' \rangle$ y $\langle \uparrow, \downarrow \rangle \mapsto I_{\langle \uparrow, \downarrow \rangle}$ son mutuamente mapeos inversos entre el conjunto de relaciones binarias entre X e Y y el conjunto de todas las conexiones de Galois entre X e Y .

Observaciones

En particular, el teorema anterior asegura que (1)-(4) describe completamente todas las propiedades de nuestros operadores inducidos por los datos $\langle X, Y, I \rangle$.

2.7.8. Estructura jerárquica de retículo de conceptos

Sabemos que $\mathcal{B}(X, Y, I)$ (conjunto de todos los conceptos formales) con \leq (jerarquía subconcepto-superconcepto) es un conjunto parcialmente ordenado. Ahora, la cuestión es: ¿Cuál es la estructura de $\langle \mathcal{B}(X, Y, I), \leq \rangle$?

Resulta que $\langle \mathcal{B}(X, Y, I), \leq \rangle$ es un retículo completo (se verá en la parte de Teorema principal de retículos de conceptos).

Retículo de conceptos \approx jerarquía de conceptos completa

Que $\langle \mathcal{B}(X, Y, I), \leq \rangle$ sea un retículo es una buena noticia.

Concretamente, se dice que para cualquier colección de conceptos formales $K \subseteq \mathcal{B}(X, Y, I)$, contiene tanto la "generalización directa" $\bigvee K$ de conceptos de K (supremo de K) , como la "especialización directa" $\bigwedge K$ de conceptos de K (ínfimo de K). En este sentido, es una jerarquía completa de conceptos.

A continuación se detalla el Teorema principal de retículos de conceptos.

Teorema 2.7.11. Para un operador de cierre C en X , el conjunto parcialmente ordenado $\langle \text{fix}(C), \subseteq \rangle$ de puntos fijos de C es un retículo completo con ínfimo y supremo dado por

$$\bigwedge_{j \in J} A_j = \bigcap_{j \in J} A_j, \quad (2.12)$$

$$\bigvee_{j \in J} A_j = C(\bigcup_{j \in J} A_j). \quad (2.13)$$

2.8. Teorema Principal de Retículo de Conceptos

Teorema 2.8.1. 1. $\mathcal{B}(X, Y, I)$ (1) es un retículo completo con ínfimo y supremo dado por

$$\bigwedge_{j \in J} \langle A_j, B_j \rangle = \left\langle \bigcap_{j \in J} A_j, (\bigcup_{j \in J} B_j)^{\uparrow\downarrow} \right\rangle = \bigvee_{j \in J} \langle A_j, B_j \rangle = \left\langle (\bigcup_{j \in J} A_j)^{\uparrow\downarrow}, (\bigcap_{j \in J} B_j) \right\rangle. \quad (2.14)$$

2. Además, un retículo completo arbitrario $\mathbf{V} = (V, \leq)$ es isomorfo a $\mathcal{B}(X, Y, I)$ si y sólo si hay mapeos $\gamma : X \rightarrow V, \mu : Y \rightarrow V$ tal que

- a) $\gamma(X)$ es \bigvee -irreducible en V , $\mu(Y)$ es \bigwedge -irreducible en V ;
- b) $\gamma(X) \leq \mu(Y)$ sii $\langle x, y \rangle \in I$.

Observaciones

1. $K \subseteq V$ es supremo-irreducible en V si y sólo si para cada $v \in V$ existe $K' \subseteq K$ tal que $v = \bigvee K'$ (es decir, todo elemento v de V es supremo de algún elemento de K).

Igualmente para el ínfimo-irreducible de K en V (todo elemento v de V es ínfimo de algún elemento de K).

2. Supremo (ínfimo)-irreducibilidad establece que puedan ser considerados bloques de trabajo de V .

¿Qué dice el Teorema Principal? La parte (1) dice que $\mathcal{B}(X, Y, I)$ es un retículo y describe su ínfimo y supremo. La parte (2) provee la forma de etiquetar un retículo de conceptos para que esa información no se pierda.

2.8.1. Etiquetado de diagramas de retículos de conceptos

El etiquetado tiene dos reglas:

- $\gamma(x) = \langle \{x\}^{\uparrow\downarrow}, \{x\}^\uparrow \rangle$... objeto del concepto de x - etiquetado para x ,
- atributo del concepto de y - etiquetado para y .

¿Cómo vemos las extensiones e intensiones en un diagrama de Hasse etiquetado?

Consideremos el concepto formal $\langle A, B \rangle$ correspondiente al nodo c de un diagrama etiquetado del retículo de conceptos $\mathcal{B}(X, Y, I)$. ¿Qué es la extensión y la intención en $\langle A, B \rangle$?

- $x \in A$ si y sólo si el nodo con la etiqueta x se encuentra en el camino desde c hacia abajo,
- $y \in B$ si y sólo si el nodo con la etiqueta y se encuentra en el camino desde c hacia arriba.

	y_1	y_2	y_3	y_4
x_1	X	X	X	X
x_2		X	X	X
x_3			X	X
x_4	X			

Figura 2.6: Contexto Formal

$$\mathcal{B}(X, Y, I) = \{\langle \{x_1\}, Y \rangle, \langle \{x_1, x_2, x_3\}, \{y_2, y_3, y_4\} \rangle, \langle \{x_1, x_4\}, \{y_1\} \rangle, \langle X, \emptyset \rangle\}$$

El diagrama correspondiente:

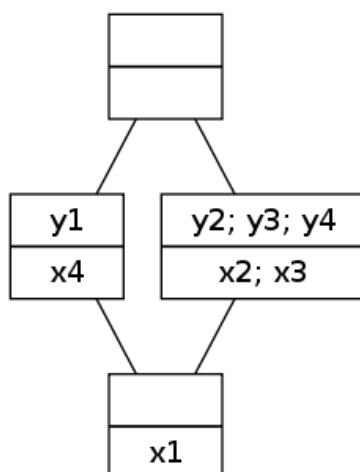


Figura 2.7: Diagrama de etiquetado del retículo de conceptos

Sea c el nodo del concepto $\langle \{x_2, x_3\}, \{y_2, y_3, y_4\} \rangle$. La extensión de c son los x_i de los nodos del camino desde c hacia abajo, es decir $\{x_2, x_3, x_1\}$.

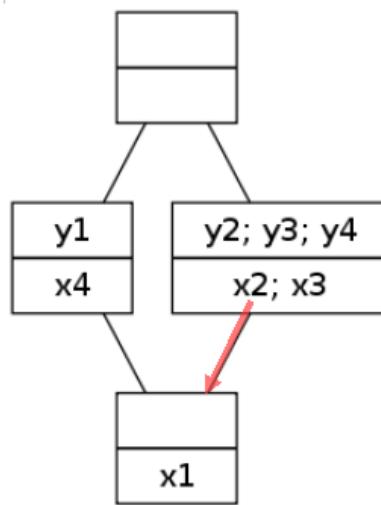


Figura 2.8: Extensión del concepto $\langle \{x_2, x_3\}, \{y_2, y_3, y_4\} \rangle$

La intención de c son los y_i de los nodos del camino desde c hacia arriba, es decir $\{y_2, y_3, y_4\}$.

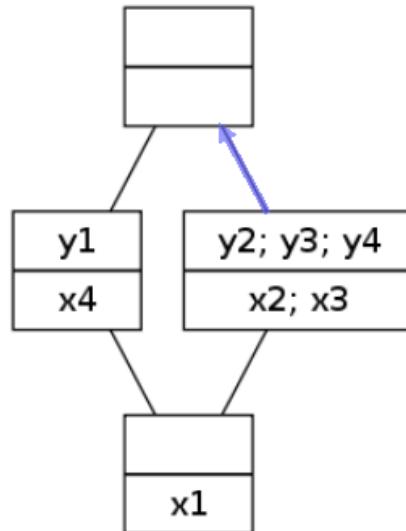


Figura 2.9: Intensión del concepto $\langle \{x_2, x_3\}, \{y_2, y_3, y_4\} \rangle$

2.9. Clarificación y reducción de conceptos formales

Un contexto formal puede ser redundante y es posible borrar algunos de sus objetos o atributos y obtener un contexto formal para el cual el retículo de conceptos asociado es isomorfo al del contexto formal original.

Para ello se definirán dos nociones principales: contexto formal clarificado y contexto formal reducido.

Definición (contexto clarificado): Un contexto formal $\langle X, Y, I \rangle$ es clarificado cuando

$$\begin{aligned}\{x_1\}^\uparrow &= \{x_2\}^\uparrow \text{ implica } x_1 = x_2 \text{ para todo } x_1, x_2 \in X; \\ \{y_1\}^\downarrow &= \{y_2\}^\downarrow \text{ implica } y_1 = y_2 \text{ para todo } y_1, y_2 \in Y.\end{aligned}$$

La clarificación consiste en la eliminación de idénticas filas y columnas (queda sólo una de las filas/columnas idénticas).

Ejemplo:

El contexto formal de la derecha resulta de aplicar la clarificación del contexto formal de la izquierda.

<i>I</i>	<i>y₁</i>	<i>y₂</i>	<i>y₃</i>	<i>y₄</i>
<i>x₁</i>	×	×	×	✗
<i>x₂</i>	✗		✗	✗
<i>x₃</i>		✗	✗	✗
<i>x₄</i>		✗	✗	✗
<i>x₅</i>	✗			

<i>I</i>	<i>y₁</i>	<i>y₂</i>	<i>y₃</i>
<i>x₁</i>	✗	✗	✗
<i>x₂</i>	✗		✗
<i>x₃</i>		✗	✗
<i>x₅</i>	✗		

Figura 2.10: Clarificación

Teorema 2.9.1. Si $\langle X_1, Y_1, I_1 \rangle$ es un contexto clarificado resultante de $\langle X_2, Y_2, I_2 \rangle$ por clarificación, entonces $\mathcal{B}(X_1, Y_1, I_1)$ es isomorfo a $\mathcal{B}(X_2, Y_2, I_2)$.

Definición (objetos y atributos reducibles):

Para un contexto formal $\langle X, Y, I \rangle$, un atributo $y \in Y$ es reducible si y sólo si existe $Y' \subset Y$ con $y \notin Y'$ tal que

$$\{y\}^\downarrow = \bigcap_{z \in Y'} \{z\}^\downarrow,$$

es decir, la columna correspondiente a y es la intersección de las columnas correspondientes a los elementos (z) de Y' . Un objeto $x \in X$ es reducible si y sólo si existe

$X' \subset X$ con $x \notin X'$ tal que

$$\{x\}^\uparrow = \bigcap_{z \in X'} \{z\}^\uparrow,$$

es decir, la fila correspondiente a x es la intersección de filas correspondientes a los elementos (z) de X' .

Ejemplo: Dado el contexto formal

	y_1	y_2	y_3
x_1			\times
x_2	\times	\times	\times
x_3	\times		

y_2 es reducible ($Y' = y_1, y_3$).

Observaciones

- Si un elemento y (valor real del atributo) es una combinación lineal de otros atributos, éste puede ser eliminado. Cuidado, esto depende de lo que se hace con los atributos. La intersección es una combinación particular de atributos.
- La (no-)reducibilidad en $\langle X, Y, I \rangle$ está conectado con la llamada \wedge -(ir)-reducibilidad y \vee -(ir)reducibilidad en $\mathcal{B}(X, Y, I)$.
- En un retículo completo $\langle \vee, \leq \rangle$ es llamado \wedge -irreducible si no hay $U \subset V$ con $v \notin U$ tal que $v = \bigwedge U$. Igualmente para \vee -irreducibilidad.
- Por definición, y es reducible si y sólo si hay $Y' \subset Y$ con tal que

$$\left\langle \{y\}^\downarrow, \{y\}^{\downarrow\uparrow} \right\rangle = \bigwedge_{z \in Y'} \left\langle \{z\}^\downarrow, \{z\}^{\downarrow\uparrow} \right\rangle \quad (2.15)$$

Sea $\langle X, Y, I \rangle$ clarificado. Entonces en (16), para cada $z \in Y'$: $\{y\}^\downarrow \neq \{z\}^\downarrow$, y por lo tanto, $\left\langle \{y\}^\downarrow, \{y\}^{\downarrow\uparrow} \right\rangle$. Por consiguiente, y es reducible si y sólo si $\left\langle \{y\}^\downarrow, \{y\}^{\downarrow\uparrow} \right\rangle$ es un ínfimo de atributos de conceptos diferente de $\left\langle \{y\}^\downarrow, \{y\}^{\downarrow\uparrow} \right\rangle$. Ahora, como todo concepto $\langle A, B \rangle$ es un ínfimo de algún atributo de conceptos (los atributos de conceptos \wedge -irreducibles), obtenemos que y no es reducible si y sólo si $\left\langle \{y\}^\downarrow, \{y\}^{\downarrow\uparrow} \right\rangle$ es \wedge -irreducible en $\mathcal{B}(X, Y, I)$.

Por lo que, si $\langle X, Y, I \rangle$ es clarificado, y no es reducible si y sólo si $\left\langle \{y\}^\downarrow, \{y\}^{\downarrow\uparrow} \right\rangle$ es \wedge -irreducible.

Supongamos que $\langle X, Y, I \rangle$ no es clarificado debido a que $\{y\}^\downarrow = \{z\}^\downarrow$ para algún $z \neq y$. Entonces se puede ver que y es reducible por definición sólo haciendo que $Y' = \{z\}$. Sin embargo, puede suceder que $\langle \{y\}^\downarrow, \{y\}^{\uparrow\downarrow} \rangle$ sea \wedge -irreducible y puede suceder que y sea \wedge -irreducible.

Ejemplo:

En la figura 2.11 se muestran dos contextos no clarificados. En la izquierda, y_2 es reducible y $\langle \{y_2\}^\downarrow, \{y_2\}^{\uparrow\downarrow} \rangle$ \wedge -reducible. En la derecha, y_2 es reducible pero $\langle \{y_2\}^\downarrow, \{y_2\}^{\uparrow\downarrow} \rangle$ \wedge -irreducible.

I	y_1	y_2	y_3	y_4
x_1			X	
x_2	X	X	X	X
x_3	X	X	X	X
x_4	X			

I	y_1	y_2	y_3	y_4	y_5
x_1	X		X		
x_2			X		X
x_3	X	X	X	X	X
x_4	X		X		

Figura 2.11: Contextos no clarificados

Igualmente para la reducibilidad de objetos. Si $\langle X, Y, I \rangle$ es clarificado, entonces x no es reducible si y sólo si $\langle \{x\}^{\uparrow\downarrow}, \{x\}^\uparrow \rangle$ es \wedge -irreducible en $\mathcal{B}(X, Y, I)$.

Por lo que, es conveniente considerar la reducibilidad en los contextos clarificados, luego, la reducibilidad de objetos y atributos corresponden a \vee - y \wedge -reducibilidad de objetos y atributos de conceptos.

Teorema 2.9.2. *Sea $y \in Y$ reducible en $\langle X, Y, I \rangle$. Entonces $\mathcal{B}(X, Y - \{y\}, J)$ es isomorfo a $\mathcal{B}(X, Y, I)$ donde $J = I \cap (X \times (Y - \{y\}))$ resulta de eliminar la columna y de $\langle X, Y, I \rangle$.*

Definición (contexto formal reducido): $\langle X, Y, I \rangle$ es

- reducido por filas si no existe ningún objeto $x \in X$ reducible,
- reducido por columnas si no existe ningún atributo $y \in Y$ reducible,
- reducido si es reducido por filas y por columnas.

Por la observación anterior, si $\langle X, Y, I \rangle$ no es clarificado, entonces o bien algún objeto es reducible (si hay filas iguales) o algún atributo es reducible (si hay columnas iguales).

Por lo que, si $\langle X, Y, I \rangle$ es reducido, éste es clarificado.

La relación entre reducibilidad de objetos/atributos y \vee - y \wedge -reducibilidad de objetos/atributos de conceptos lleva a la siguiente conclusión:

Un clarificado $\langle X, Y, I \rangle$ es

- reducido por filas si y sólo si todos los objetos de concepto son \vee -irreducible,

- reducido por columnas si y sólo si todos los atributos de concepto son \wedge -irreducible,

2.9.1. Reducción de contextos formales por relaciones de vectores

¿Cómo averiguar qué objetos y atributos son reducibles?

Definición:

Para $\langle X, Y, I \rangle$, se definen las relaciones $\nearrow \swarrow \uparrow \downarrow$ entre X e Y por

- $x \swarrow y$ si $\langle x, y \rangle \notin I$ y si $\{x\}^\uparrow \subset \{x_1\}^\uparrow$ entonces $\langle x_1, y \rangle \in I$.
- $x \nearrow y$ si $\langle x, y \rangle \notin I$ y si $\{y\}^\downarrow \subset \{y_1\}^\downarrow$ entonces $\langle x, y_1 \rangle \in I$.
- $x \uparrow y$ si $x \swarrow y$ y $x \nearrow y$.

Por lo que, si $\langle x, y \rangle \in I$ entonces no ocurre $x \nearrow y, x \swarrow y, x \uparrow y$. Las relaciones de vectores pueden por lo tanto ser introducidas en la tabla de $\langle X, Y, I \rangle$ como en el siguiente ejemplo:

I	y_1	y_2	y_3	y_4
x_1	✗	✗	✗	✗
x_2	✗	✗		
x_3		✗	✗	✗
x_4	✗			
x_5	✗	✗		

I	y_1	y_2	y_3	y_4
x_1	✗	✗	✗	✗
x_2	✗	✗	✗	✗
x_3	✗	✗	✗	✗
x_4	✗	✗	✗	✗
x_5	✗	✗	✗	✗

Figura 2.12: Representación de relaciones de vectores

Teorema 2.9.3. (*las relaciones de vectores y la reducibilidad*) Para cualquier $\langle X, Y, I \rangle$, $x \in X, y \in Y$:

- $\langle \{x\}^{\uparrow\downarrow}, \{x\}^\uparrow \rangle$ es \vee -irreducible si existe $y \in Y$ tal que $x \swarrow y$;
- $\langle \{x\}^{\uparrow\downarrow}, \{x\}^\downarrow \rangle$ es \wedge -irreducible si existe $x \in X$ tal que $x \nearrow y$;

Considérese el siguiente problema:

INPUT: Contexto formal arbitrario $\langle X_1, Y_1, I_1 \rangle$

OUTPUT: Un contexto reducido $\langle X_2, Y_2, I_2 \rangle$

Este problema se puede resolver con el siguiente algoritmo:

1. clarificar $\langle X_1, Y_1, I_1 \rangle$ para obtener un contexto clarificado $\langle X_3, Y_3, I_3 \rangle$, borrando las filas y columnas iguales,
2. calcular las relaciones de vectores \nearrow y \swarrow para $\langle X_3, Y_3, I_3 \rangle$,
3. obtener $\langle X_2, Y_2, I_2 \rangle$ a partir de $\langle X_3, Y_3, I_3 \rangle$ borrando los objetos x de X_3 para los que no existe $y \in Y_3$ con $x \swarrow y$, y atributos y de Y_3 para los que no existe $x \in X_3$ con $x \nearrow y$. Esto es:

$$X_2 = X_3 - \{x \mid \text{no existe } y \in Y_3 \text{ tal que } x \swarrow y\},$$

$$Y_2 = Y_3 - \{y \mid \text{no existe } x \in X_3 \text{ tal que } x \nearrow y\},$$

$$I_2 = I_3 \cap (X_2 \times Y_2).$$

Ejemplo:

Calcular las relaciones de vectores \nearrow , \swarrow y \uparrow para el siguiente contexto formal:

l_1	y_1	y_2	y_3	y_4
x_1	X	X	X	X
x_2	X	X		
x_3		X	X	
x_4		X		
x_5		X	X	

Figura 2.13: Contexto de entrada

Empezamos con \nearrow . Necesitamos recorrer las celdas en la tabla que no contengan \times y decidir si aplicar \nearrow .

La primera de dichas celdas corresponde a $\langle x_2, y_3 \rangle$. Por definición $x_2 \nearrow y_3$ si para cada $y \in Y$ tal que $\{y_3\}^\downarrow \subset \{y\}^\downarrow$ tenemos $x_2 \in \{y\}^\downarrow$. El único y que cumple la condición es y_2 para el cual tenemos $x_2 \in \{y_2\}^\downarrow$, de ahí que $x_2 \nearrow y_3$.

Y así sucesivamente hasta $\langle x_5, y_4 \rangle$ para los que obtenemos $x_5 \nearrow y_4$.

Continuamos con \swarrow . Recorremos las celdas de la tabla que no contienen X y decidimos si aplicar \swarrow . La primera de dichas celdas corresponde a $\langle x_2, y_3 \rangle$. Por definición, $x_2 \swarrow y_3$ si para cada $x \in X$ tal que $\{x_2\}^\uparrow \subset \{x\}^\uparrow$. El único x que cumple la condición es x_1 para el que tenemos que $y_3 \in \{x\}^\uparrow$, de ahí que $x_2 \swarrow y_3$.

Y así sucesivamente hasta $\langle x_5, y_4 \rangle$ para los que obtenemos $x_5 \swarrow y_4$.

Las relaciones de vectores se muestran en la siguiente tabla:

l_1	y_1	y_2	y_3	y_4
x_1	X	X	X	X
x_2	X	X	‡	↙
x_3	‡	X	X	X
x_4	↗	X	↗	
x_5	↗	X	X	‡

Figura 2.14: Cálculo de relación de vectores

Por lo que, el correspondiente contexto reducido es

l_1	y_1	y_3	y_4
x_2	X		
x_3		X	X
x_5		X	

Figura 2.15: Contexto reducido

Para un retículo completo $\langle V, \leq \rangle$ y $v \in V$, señalar que

$$v_* = \bigvee_{u \in V, u < v} u,$$

$$v^* = \bigwedge_{u \in V, u < v} u.$$

Sea $\langle X_1, Y_1, I_1 \rangle$ clarificado , $X_2 \subseteq X_1$ y conjuntos de objetos y atributos irreducibles, respectivamente, sea $I_2 = I_1 \cap (X_2 \times Y_2)$ (restricción de I_1 a los objetos y atributos irreducibles).

¿Cómo podemos obtener a partir de los conceptos de $\mathcal{B}(X_1, Y_1, I_1)$ los conceptos de $\mathcal{B}(X_2, Y_2, I_2)$? La respuesta está basada en:

1. $\langle A_1, B_1 \rangle \mapsto \langle A_1 \cap X_2, B_1 \cap Y_2 \rangle$ es un isomorfismo de $\mathcal{B}(X_1, Y_1, I_1)$ en $\mathcal{B}(X_2, Y_2, I_2)$.
2. Por lo tanto, cada extensión A_2 de $\mathcal{B}(X_1, Y_1, I_1)$ es de la forma $A_2 = A_1 \cap X_2$ donde A_1 es una extensión de $\mathcal{B}(X_1, Y_1, I_1)$ (igual para las intensiones).
3. Para $x \in X_1 : x \in A_1$ si $\{x\}^{\uparrow\downarrow} \cap X_2 \subseteq A_1 \cap X_2$, para $y \in Y_1 : y \in B_1$ si $\{y\}^{\uparrow\downarrow} \cap Y_2 \subseteq B_1 \cap Y_2$.

Aquí, \uparrow y \downarrow son operadores inducidos por $\langle X_1, Y_1, I_1 \rangle$. Por lo que, dado $\langle A_2, B_2 \rangle \in \mathcal{B}(X_2, Y_2, I_2)$, el correspondiente $\langle A_1, B_1 \rangle \in \mathcal{B}(X_1, Y_1, I_1)$ es dado por

$$A_1 = A_2 \cup \left\{ x \in X_1 - X_2 \mid \{x\}^{\uparrow\downarrow} \cap X_2 \subseteq A_2 \right\}, \quad (2.16)$$

$$B_1 = B_2 \cup \left\{ y \in Y_1 - Y_2 \mid \{y\}^{\uparrow\downarrow} \cap Y_2 \subseteq B_2 \right\}, \quad (2.17)$$

Ejemplo

A la izquierda un contexto formal clarificado $\langle X_1, Y_1, I_1 \rangle$, a la derecha un contexto reducido $\langle X_2, Y_2, I_2 \rangle$ (ver el ejemplo anterior).

I_1	y_1	y_2	y_3	y_4
x_1	×	×	×	×
x_2	×	×		
x_3		×	×	×
x_4		×		
x_5		×	×	

I_2	y_1	y_3	y_4
x_2	×		
x_3		×	×
x_5		×	

Figura 2.16: Contextos de entrada

Vamos a determinar $\mathcal{B}(X_1, Y_1, I_1)$ primero calculando $\mathcal{B}(X_2, Y_2, I_2)$ y luego usando el método descrito anteriormente para obtener conceptos $\mathcal{B}(X_1, Y_1, I_1)$ de los correspondientes conceptos de $\mathcal{B}(X_2, Y_2, I_2)$.

$\mathcal{B}(X_2, Y_2, I_2)$ está formado por:

$$\langle \emptyset, Y_2 \rangle, \langle \{x_2\}, \{y_1\} \rangle, \langle \{x_3\}, \{y_3, y_4\} \rangle, \langle \{x_3, x_5\}, \{y_3\} \rangle, \langle X_2, \emptyset \rangle$$

Necesitamos recorrer todos los $\langle A_2, B_2 \rangle \in \mathcal{B}(X_2, Y_2, I_2)$ y determinar el correspondiente $\langle A_1, B_1 \rangle \in \mathcal{B}(X_1, Y_1, I_1)$ usando (17) y (18). Nótese: $X_1 - X_2 = \{x_1, x_4\}$, $Y_1 - Y_2 = \{y_2\}$.

1. para $\langle A_2, B_2 \rangle = \langle \emptyset, Y_2 \rangle$ tenemos

$$\{x_1\}^{\uparrow\downarrow} \cap X_2 = \{x_1\} \cap X_2 = \emptyset \subseteq A_2,$$

$$\{x_4\}^{\uparrow\downarrow} \cap X_2 = X_1 \cap X_2 = X_2 \not\subseteq A_2,$$

de ahí que $A_1 = A_2 \cup \{x_1\} = \{x_1\}$, y

$$\{y_2\}^{\uparrow\downarrow} \cap Y_2 = \{y_2\} \cap Y_2 = \emptyset \subseteq B_2,$$

por lo que $B_1 = B_2 \cup \{y_2\} = Y_1$. Por eso, $\langle A_1, B_1 \rangle = \langle \{x_1\}, Y_1 \rangle$.

2. para $\langle A_2, B_2 \rangle = \langle \{x_2\}, \{y_1\} \rangle$ tenemos

$$\{x_1\}^{\uparrow\downarrow} \cap X_2 = \emptyset \subseteq A_2, \{x_4\}^{\uparrow\downarrow} \cap X_2 \not\subseteq A_2,$$

de ahí que $A_1 = A_2 \cup \{x_1\} = \{x_1, x_2\}$, y

$$\{y_2\}^{\uparrow\downarrow} \cap Y_2 = \{y_2\} \cap Y_2 = \emptyset \subseteq B_2,$$

por lo que $B_1 = B_2 \cup \{y_2\} = \{y_1, y_2\}$. Por eso, $\langle A_1, B_1 \rangle = \langle \{x_1, x_2\}, \{y_1, y_2\} \rangle$.

3. para $\langle A_2, B_2 \rangle = \langle \{x_3\}, \{y_3, y_4\} \rangle$ tenemos

$$\{x_1\}^{\uparrow\downarrow} \cap X_2 = \emptyset \subseteq A_2, \{x_4\}^{\uparrow\downarrow} \cap X_2 = X_2 \not\subseteq A_2,$$

de ahí que $A_1 = A_2 \cup \{x_1\} = \{x_1, x_3\}$ y

$$\{y_2\}^{\uparrow\downarrow} \cap Y_2 = \{y_2\} \cap Y_2 = \emptyset \subseteq B_2,$$

por lo que $B_1 = B_2 \cup \{y_2\} = \{y_2, y_3, y_4\}$. Por eso, $\langle A_1, B_1 \rangle = \langle \{x_1, x_3\}, \{y_2, y_3, y_4\} \rangle$.

4. para $\langle A_2, B_2 \rangle = \langle \{x_3, x_5\}, \{y_3\} \rangle$ tenemos

$$\{x_1\}^{\uparrow\downarrow} \cap X_2 = \emptyset \subseteq A_2, \{x_4\}^{\uparrow\downarrow} \cap X_2 = X_2 \not\subseteq A_2,$$

de ahí que $A_1 = A_2 \cup \{x_1\} = \{x_1, x_3, x_5\}$ y

$$\{y_2\}^{\uparrow\downarrow} \cap Y_2 = \{y_2\} \cap Y_2 = \emptyset \subseteq B_2,$$

por lo que $B_1 = B_2 \cup \{y_2\} = \{y_2, y_3\}$. Por eso, $\langle A_1, B_1 \rangle = \langle \{x_1, x_3, x_5\}, \{y_2, y_3\} \rangle$.

5. para $\langle A_2, B_2 \rangle = \langle X_2, \emptyset \rangle$ tenemos

$$\{x_1\}^{\uparrow\downarrow} \cap X_2 = \emptyset \subseteq A_2, \{x_4\}^{\uparrow\downarrow} \cap X_2 = X_2 \subseteq A_2,$$

de ahí que $A_1 = A_2 \cup \{x_1, x_4\} = X_1$, y

$$\{y_2\}^{\uparrow\downarrow} \cap Y_2 = \{y_2\} \cap Y_2 = \emptyset \subseteq B_2,$$

por lo que $B_1 = B_2 \cup \{y_2\} = \{y_2\}$. Por eso, $\langle A_1, B_1 \rangle = \langle X_1, \{y_2\} \rangle$.

2.10. Algoritmos para el cálculo de retículos de conceptos

Consideremos el problema del cálculo de retículos de conceptos, es decir, el siguiente problema:

INPUT: contexto formal $\langle X, Y, I \rangle$, OUTPUT: retículo de conceptos $\mathcal{B}(X, Y, I)$ (posiblemente más \leq)

- A veces se necesita calcular el conjunto $\mathcal{B}(X, Y, I)$ del concepto formal sólo.
- A veces se necesita calcular el conjunto $\mathcal{B}(X, Y, I)$ y la jerarquía conceptual \leq . \leq puede ser calculado a partir de $\mathcal{B}(X, Y, I)$ por definición de \leq . Pero eso no es eficiente. Existen algoritmos que pueden calcular $\mathcal{B}(X, Y, I)$ y \leq simultáneamente, lo cual es más eficiente que primero calcular $\mathcal{B}(X, Y, I)$ y luego \leq .

En los siguientes apartados describiremos los algoritmos "Next-Closure" de Ganter y el del "Vecino superior" de Lindig.

2.10.1. Algoritmo Next-Closure

Autor: Bernhard Ganter (1987)

Entrada: contexto formal $\langle X, Y, I \rangle$,

Salida: $Int(X, Y, I) \dots$ todas las intensiones (igualmente, $Ext(X, Y, I) \dots$ todas las extensiones).

Lista todas las intensiones (o extensiones) en orden lexicográfico. Nótese que $\mathcal{B}(X, Y, I)$ puede ser reconstruido a partir de $Int(X, Y, I)$ debido a

$$\mathcal{B}(X, Y, I) = \left\{ \langle B^\downarrow, B \rangle \mid B \int Int(X, Y, I) \right\}$$

Es uno de los algoritmos más populares y fácil de implementar.

Se describe Next-Closure para intensiones.

Supongamos $Y = \{1, \dots, n\}$, esto es, denotamos los atributos con enteros positivos, de esta manera, fijamos un orden de atributos.

Definición

Para $A, B \subseteq Y, i \in 1, \dots, n$

$$A <_i B \text{ si } i \in B - A \text{ y } A \cap \{1, \dots, i-1\} = B_{cap} \{1, \dots, i-1\},$$

$$A < B \text{ si } A <_i \text{ para algún } i$$

Nota: ¡... orden lexicográfico, esto es, todo par de conjuntos distintos $A, B \subseteq$ son comparables.

Para $i = 1$, establecemos $\{1, \dots, i-1\} = \emptyset$.

Podríamos pensar en $B \subseteq Y$ en términos de sus características de vector. Para $Y = \{1, 2, 3, 4, 5, 6, 7\}$ y $B = \{1, 3, 4, 6\}$, la característica de vector de \mathcal{B} es 1011010.

Definición

Para $A \subseteq Y, i \in \{1, \dots, n\}$

$$A \oplus i := ((A \cap \{1, \dots, i-1\}) \cup \{i\})^{\downarrow\uparrow}$$

Ejemplo

<i>I</i>	1	2	3	4
<i>x₁</i>	×	×		×
<i>x₂</i>	×	×	×	×
<i>x₃</i>		×		

- $A = \{1, 3\}, i = 2.$

$$A \oplus i = ((\{1, 3\} \cap \{1, 2\}) \cup \{2\})^{\downarrow\uparrow} = (\{1\} \cup \{2\})^{\downarrow\uparrow} = \{1, 2\}^{\downarrow\uparrow} = \{1, 2, 4\}$$

- $A = 2, i = 1.$

$$A \oplus i = ((\{2\} \cap \emptyset) \cup \{1\})^{\downarrow\uparrow} = \{1\}^{\downarrow\uparrow} = \{1, 2, 4\}$$

Lema

Para cualquier $B, D, D_1, D_2 \subseteq Y$:

- (1) Si $B <_i D_1, B <_j D_2$, y $i < j$ entonces $D_2 <_i D_1$;
- (2) Si $i \notin B$ entonces $B < B \oplus i$;
- (3) Si $B <_i D$ y $D = D^{\downarrow\uparrow}$ entonces $B \oplus i \subseteq D$;
- (4) Si $B <_i D$ y $D = D^{\downarrow\uparrow}$ entonces $B <_i B \oplus i$.

Teorema 2.10.1. La menor intensión B^+ mayor que $B \subseteq Y$ es dada por

$$B^+ = B \oplus i$$

donde i es el mayor elemento con $B <_i B \oplus i$.

Pseudo-código del algoritmo Next-Closure

1. $A := \emptyset^{\downarrow\uparrow}; \text{ (leastIntent)}$
2. $\text{store}(A);$
3. $\text{while not } (A = Y) \text{ do}$
4. $A := A+;$
5. $\text{store}(A);$
6. endwhile.

complejidad: la complejidad de tiempo de cálculo de A^+ es $O(|X| \cdot |Y|^2)$; la complejidad de cálculo de C^\uparrow es $O(|X| \cdot |Y|)$; la complejidad de cálculo de $A \oplus i$ es por tanto $O(|X| \cdot |Y|)$. Para obtener A^+ necesitamos calcular $A \oplus i |Y|$ -veces en el peor de los casos. Como resultado, la complejidad de calcular A^+ es $O(|X| \cdot |Y|^2)$.

La complejidad de tiempo de Next-Closure es $O(|X| \cdot |Y|^2 \cdot \mathcal{B}(X, Y, I))$.

complejidad de retardo de tiempo polinómico N: el método para calcular A^+ desde A tiene complejidad polinómica, por tanto, el algoritmo Next-Closure tiene un tiempo de retardo polinómico.

Observaciones

- Si $\downarrow\uparrow$ es sustituido por un operador de cierre C arbitrario, el algoritmo Next-Closure calcula todos los puntos fijos de C .
- Por lo que, el algoritmo Next-Closure es esencialmente un algoritmo para el cálculo de todos los puntos fijos de un operador de cierre C dado.
- La complejidad computacional de Next-Closure depende de la complejidad computacional del cálculo de $C(A)$ (cálculo de cierre de un conjunto arbitrario A).

2.10.2. Algoritmo UpperNeighbor

Autor: Christian Lindig (Fast Concept Analysis, 2000)

Entrada: Contexto formal $\langle X, Y, I \rangle$

Salida: $\mathcal{B}(X, Y, I)$ y \leq

La idea básica del algoritmo es:

1. comenzar con el menor concepto formal $\langle \emptyset^{\uparrow\downarrow}, \emptyset^\uparrow \rangle$,
2. para cada $\langle A, B \rangle$ generar todos los vecinos superiores y guardar la información necesaria
3. ir al próximo concepto

El punto crucial es cómo calcular los vecinos superiores de un $\langle A, B \rangle$ dado.

Teorema 2.10.2. Si $\langle A, B \rangle \in \mathcal{B}(X, Y, I)$ no es el concepto más grande entonces $(A \cup \{x\})^{\uparrow\downarrow}$, con $x \in X - A$, es una extensión de un vecino superior de $\langle A, B \rangle$ si para cada $z \in (A \cup \{x\})^{\uparrow\downarrow} - A$ tenemos $(A \cup \{x\})^{\uparrow\downarrow} = (A \cup \{z\})^{\uparrow\downarrow}$.

Observaciones

En general, para $x \in X - A$, $(A \cup \{x\})^{\uparrow\downarrow}$ no tiene que ser una extensión de un vecino superior de $\langle A, B \rangle$.

Pseudo-código del algoritmo UpperNeighbor

```

1. min := X - A;
2. neighbors := ∅;
3. for x ∈ X - A do
4.    $B_1 := (A \cup \{x\})^{\uparrow}; A_1 := B_1^{\downarrow};$ 
5.   if  $(min \cap ((A_1 - A) - \{x\})) = \emptyset$  then
6.     neighbors := neighbors ∪  $\{(A_1, B_1)\}$ 
7.   else min := min - {x};
8. enddo.
```

complejidad: tiempo de retardo polinómico con retardo $O(|X|^2 \cdot |Y|)$ (igual que la versión para las extensiones del algoritmo Next-Closure).

2.11. Contextos multivaluados y escalamiento conceptual

Un contexto $\langle X, Y, I \rangle$ es multivaluado, cuando los elementos de I pueden tener más de dos valores. Por ejemplo:

	age	education	symptom
Alice	23	BS	1
Boris	30	MS	0
Cyril	31	PhD	1
David	43	MS	0
Ellen	24	PhD	1
Fred	64	MS	0
George	30	Bc	0

Figura 2.17: Contexto multivaluado

Formalmente, un contexto multivaluado es una tupla $\mathcal{D} = \langle X, Y, W, I \rangle$ donde X es un conjunto finito no vacío de objetos, Y es un conjunto finito de atributos, W es un conjunto de valores e I es una relación ternaria entre X, Y y W que se define como:

$$I \subseteq X \times Y \times W : \langle x, y, w \rangle \in I, \langle x, y, v \rangle \in I \Rightarrow w = v$$

- Un contexto multivaluado puede representarse como una tabla cuyas filas son $x \in X$, columnas $y \in Y$ y la intersección de la fila x y la columna y contiene valores tomados de $\langle x, y, w \rangle \in I$. Si no existe la relación $\langle x, y, w \rangle \in I$, la intersección de la fila x con la columna y contendrá un espacio en blanco.
- Se puede ver que $y \in Y$ puede considerarse una función parcial de X a W . Por lo tanto, escribimos $y(x) = w$ en lugar de $\langle x, y, w \rangle \in I$.

Al conjunto $dom(y) = \{x \in X : \langle x, y, w \rangle \in I\}$ para algún $w \in W$ se le llama dominio de y . El atributo es llamado completo si $dom(y) = X$, es decir, si la tabla contiene algún valor en todas las filas de la columna correspondiente a y . Un contexto multivaluado es llamado *completo* si cada uno de sus atributos es *completo*.

- Desde el punto de vista de la teoría de bases de datos relacionales, un contexto multivaluado completo es básicamente una relación en el esquema relacional Y . Concretamente, para cada $y \in Y$ puede considerarse un atributo en el sentido de las bases de datos relacionales y afirmar que

$$D_y = \{w : \langle x, y, w \rangle \in I \text{ para algún } x \in X\}, D_y \text{ es el dominio de } y.$$

Ejemplo: Consideremos el contexto multivaluado representado en la siguiente figura:

	age	education	symptom
Alice	23	BS	1
Boris	30	MS	0
Cyril	31	PhD	1
David	43	MS	0
Ellen	24	PhD	1
Fred	64	MS	0
George	30	Bc	0

Figura 2.18: Contexto multivaluado

donde:

- $X = \{\text{Alice, Boris, ..., George}\}$

- $Y = \{\text{age}, \text{education}, \text{symptom}\}$
- $W = \{0, 1, \dots, 150, \text{BS}, \text{MS}, \text{PhD}, 0, 1\}$
- $\langle \text{Alice}, \text{age}, 23 \rangle \in I, \langle \text{Alice}, \text{education}, \text{BS} \rangle \in I, \dots, \langle \text{George}, \text{symptom}, 0 \rangle \in I$

Con lo que tenemos que, $\text{age}(\text{Alice}) = 23, \text{education}(\text{Alice}) = \text{BS}, \text{symptom}(\text{George}) = 0$.

2.11.1. Escalamiento conceptual

Para poder usar FCA como entrada contextos multivaluado éstos deben ser procesados para convertirlos en contextos formales con valores simples. A este proceso de le llama *escalamiento conceptual*.

Para ello, para cada atributo $y \in Y$ debe definirse una escala de sus valores.

Si $\langle X, Y, W, I \rangle$ es un contexto multivaluado, una escala para un atributo $y \in Y$ es un contexto formal $S_y = \langle X_y, Y_y, I_y \rangle$ tal que $D_y \subseteq X_y$. Los objetos $w \in X_y$ son llamados valores de escala, los atributos de Y_y son llamados atributos de escala.

Ejemplo:

	e_{BS}	e_{MS}	e_{PhD}
BS	1	0	0
MS	0	1	0
PhD	0	0	1

Figura 2.19: Escala para el atributo *education*

La tabla de la figura es una escala para el atributo $y = \text{education}$. Aquí $S_y = \langle X_y, Y_y, I_y \rangle, X_y = \{\text{BS}, \text{MS}, \text{PhD}\}, Y_y = \{e_{\text{BS}}, e_{\text{MS}}, e_{\text{PhD}}\}, I_y$ son los valores de la tabla.

Ejemplo:

	a_y	a_m	a_0
0	1	0	0
⋮	1	0	0
30	1	0	0
31	0	1	0
⋮	0	1	0
60	0	1	0
61	0	0	1
⋮	0	0	1
150	0	0	1

	a_y	a_m	a_0
0-30	1	0	0
31-60	0	1	0
61-150	0	0	1

Figura 2.20: Escala para el atributo *age*

La figura 2.20 es una escala para el atributo age (la tabla de la derecha es una versión abreviada de la de la izquierda). Aquí, $S_y = \langle X_y, Y_y, I_y \rangle$, $X_y = \{0, \dots, 150\}$, $Y_y = \{a_y, a_m, a_o\}$, I_y son los valores de la tabla.

Ejemplo: Una escala diferente para el atributo age es:

	a_{vy}	a_y	a_m	a_o	a_{vo}
0-25	1	0	0	0	0
26-35	0	1	0	0	0
36-55	0	0	1	0	0
56-75	0	0	0	1	0
76-150	0	0	0	0	1

Figura 2.21: Escala para el atributo age

$a_{vy} \dots$ muy joven, $a_y \dots$ joven, $a_m \dots$ mediana edad, $a_o \dots$ mayor, $a_{vo} \dots$ muy mayor.

La elección la hace el usuario y depende del nivel de precisión deseado.

Los dos tipos de escalas más importantes son:

- **Escalamiento nominal:** Los valores del atributo y no se ordenan de forma natural o no se desea tener ese orden en cuenta, es decir, y es una variable nominal.
- **Escalamiento ordinal:** Los valores del atributo y se ordenan de forma natural, es decir, y es una variable ordinal.

Ejemplo

	e_{BS}	e_{MS}	e_{PhD}
BS	1	0	0
MS	0	1	0
PhD	0	0	1

	e_{BS}	e_{MS}	e_{PhD}
BS	1	0	0
MS	1	1	0
PhD	1	1	1

Figura 2.22: Izquierda: escala nominal. Derecha: escala ordinal.

En la figura 2.22, a la izquierda se muestra una escala nominal para el atributo $y = education$. A la derecha se muestra una escala ordinal para el atributo $y=education$ con $BS \leq MS \leq PhD$.

El escalamiento conceptual define el significado de una escala de atributos de Y_y .

Aplicando dicho proceso al ejemplo de la figura 2.17, la tabla se podría transformar en:

	a_y	a_m	a_o	e_{BS}	e_{MS}	e_{PhD}	symptom
Alice	1	0	0	1	0	0	1
Boris	1	0	0	0	1	0	0
Cyril	0	1	0	0	0	1	1
David	0	1	0	0	1	0	0
Ellen	1	0	0	0	0	1	1
Fred	0	0	1	0	1	0	0
George	1	0	0	1	0	0	0

Figura 2.23: Contexto con escalamiento conceptual aplicado: contexto derivado

Si nos fijamos en la figura anterior:

- Han sido insertados nuevos atributos: a_y ... young, a_m ... middle-aged, a_o ... old, e_{BS} ... licenciado en Ciencias, e_{MS} ... tiene máster en Ciencias, e_{PhD} ... Doctor en Filosofía.
- Después del escalamiento, la información puede ser procesada mediante FCA.

El contexto formal derivado es obtenido a partir del contexto multivaluado de partida y las escalas de cada uno de sus atributos. A este proceso se le llama *escalamiento simple*.

Asumamos que $Y_{y_1} \cap Y_{y_2} = \emptyset$ para $y_1, y_2 \in Y$. Entonces, para un contexto multivaluado $\mathcal{D} = \langle X, Y, W, I \rangle$, escalas $S_y (y \in Y)$, el contexto formal derivado es $\langle X, Z, J \rangle$ donde:

- $Z = U_{y \in Y} Y y \in Y$,
- $\langle x, z \rangle \in J$ si $y(x) = w \wedge \langle w, z \rangle \in I_y$

$\langle X, Y, W, I \rangle \rightarrow \langle X, Z, J \rangle$ significa que:

- los objetos del contexto derivado son los mismos que los del contexto multivaluado original.
- cada columna que representa un atributo y es reemplazada por columnas que representan a escala de atributos $z \in Y_y$.
- El valor $y(x)$ es reemplazado por la fila del contexto de escala S_y .

Ejemplo: En la siguiente figura, se muestra el contexto formal y las escalas nominales para los atributos *age* y *education*.

	age	education	symptom
Alice	23	BS	1
Boris	30	MS	0
Cyril	31	PhD	1
David	43	MS	0
Ellen	24	PhD	1
Fred	64	MS	0
George	30	Bc	0

	a_y	a_m	a_o
0-30	1	0	0
31-60	0	1	0
61-150	0	0	1

	e_{BS}	e_{MS}	e_{PhD}
BS	1	0	0
MS	0	1	0
PhD	0	0	1

Figura 2.24: Contexto formal y escalas nominales para *age* y *education*

El contexto formal derivado es:

	a_y	a_m	a_o	e_{BS}	e_{MS}	e_{PhD}	symptom
Alice	1	0	0	1	0	0	1
Boris	1	0	0	0	1	0	0
Cyril	0	1	0	0	0	1	1
David	0	1	0	0	1	0	0
Ellen	1	0	0	0	0	1	1
Fred	0	0	1	0	1	0	0
George	1	0	0	1	0	0	0

Figura 2.25: Contexto formal derivado

Capítulo 3

ALGORITMOS DE CÁLCULO DE BASES

En esta sección se incluyen los pseudocódigos de los algoritmos implementados para el cálculo de bases directas, por el grupo de investigación GIMAC, en colaboración con la Dra. K. Bertet de la Universidad de La Rochelle. Uno de los objetivos de este proyecto es implementar estos algoritmos en Java siguiendo las estructuras de datos que están implementadas en el repositorio <https://github.com/kbertet/java-lattices> ya que es probable que en un futuro dichos algoritmos se incluyan en esta librería.

La estructura principal usada es la clase *ImplicationalSystem* que representa un conjunto o sistema de implicaciones e implementa las operaciones clásicas sobre éstos: sistema propio, base canónica, base canónica directa, sistema unario, etc.

ImplicationalSystem está compuesto por un conjunto de elementos (atributo `set` en la Figura 3.1) y un conjunto de implicaciones representadas mediante la clase *Rule*. Ésta a su vez está compuesta por dos conjuntos de atributos que representan la premisa y la conclusión de la implicación e implementa métodos para la obtención de cada una de las partes así como para la adición y eliminación de elementos de éstas. En la Figura 3.1 se muestra el diagrama de clases con los métodos y relaciones más relevantes:

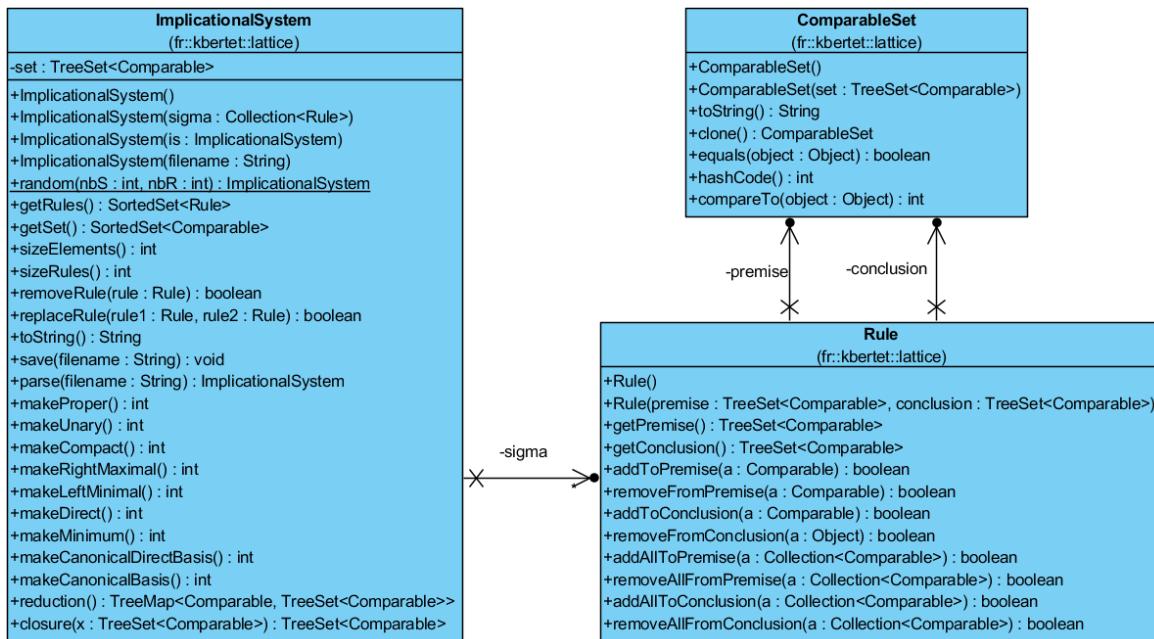


Figura 3.1: Clases librería java-lattice

Un ejemplo del uso de estos tipos, se puede ver en el cálculo de la cardinalidad de un sistema implicacional:

```

/**
 * Number of system implications.
 * @param system Implicational System.
 * @return Implications number. {@code null} when the system is null.
 */
public static final Integer getCardinality(ImplicationalSystem system) {
    Integer cardinality = null;
    if (system != null) {
        cardinality = system.sizeRules();
    }
    return cardinality;
}

```

Figura 3.2: Cálculo de la cardinalidad de un sistema implicacional

O en el cálculo de su tamaño:

```

/**
 * Sum of the number of attributes of right and left side for all implications.
 * @param system Implicational system.
 * @return The sum of the number of attributes of right and left side for all
 * implications.
 * <br/> {@code null} when the system is null.
 */
public static final Integer getSize(ImplicationalSystem system) {
    Integer size = null;

    if (system != null) {
        size = 0;
        for (Rule implication : system.getRules()) {
            size += implication.getPremise().size() + implication.getConclusion().size();
        }
    }
    return size;
}

```

Figura 3.3: Cálculo del tamaño de un sistema implicacional

3.1. CLA

A continuación se muestra el algoritmo CLA presentado en el artículo [9], definiéndose primeramente las funciones *Direct-Reduced* y *RD-Simplify* que después son usadas en el algoritmo. Para abreviar, se hará referencia al sistema implicacional con las siglas en inglés IS (Implicational System).

La función *Direct-Reduced*(Σ) calcula el IS directo-reducido de Σ :

Direct-Reduced(Σ)

input: Un sistema implicacional Σ en S
output: El IS directo-reducido Σ_{dr} en S
begin
foreach $A \rightarrow B \in \Sigma_{dr}$ y $C \rightarrow D \in \Sigma_{dr}$ **do**
if $B \cap c \neq \emptyset \neq D \setminus (A \cup B)$ **then** add $AC - B \rightarrow D - (AB)$ to Σ_{dr} ;
return Σ_{dr}

La función *RD-Simplify*(Σ) calcula el IS directo-reducido-simplificado a partir de Σ reducido:

RD-Simplify(Σ)

input: Un sistema implicacional directo-reducido Σ en S
output: El IS directo-reducido-simplificado Σ_{drs} en S equivalente a Σ
begin
 $\Sigma_{drs} := \emptyset$
foreach $A \rightarrow B \in \Sigma$ **do**
foreach $C \rightarrow D \in \Sigma$ **do**
if $C = A$ **then** $B := B \cup D$;
if $C \not\subseteq A$ **then** $B := B \setminus D$;
if $B \neq \emptyset$ **then** add $A \rightarrow B$ to Σ_{drs} ;
return Σ_{drs}

La función *doSimp*(Σ) calcula el IS directo-óptimo equivalente a Σ usando las dos funciones anteriores:

doSimp(Σ)

input: Un sistema implicacional Σ en S

output: El IS directo-óptimo Σ_{do} en S

begin

$$\Sigma_r := \{A \rightarrow B - A \rightarrow B \in \Sigma, B \not\subseteq A\}$$

$$\Sigma_{dr} := \text{Direct-Reduced}(\Sigma_r)$$

$$\Sigma_{do} := \text{RD-Simplify}(\Sigma_{dr})$$

return Σ_{do}

3.2. Direct Optimal Basis

A continuación se muestra el algoritmo Direct Optimal Basis presentado en el artículo [1]. La diferencia con el algoritmo CLA, es que se hace una simplificación izquierda y derecha antes de aplicar la regla Strong Simplification.

```

input: Un sistema implicacional  $\Sigma$  en S
output: El IS directo-óptimo  $\Sigma_{do}$  en S

1 begin
2 /* Fase 1: Generación de  $\Sigma_r$  por reducción de  $\Sigma^*$ /
3  $\Sigma_r = \emptyset$ 
4 foreach  $A \rightarrow_{\Sigma} B$  do
5   if  $B \not\subseteq A$  then add  $A \rightarrow B - A$  to  $\Sigma_r$ ;
6 /* Fase 2: Generación de  $\Sigma_{sr}$  por simplificación de  $\Sigma_r^*$ /
7  $\Sigma_{sr} = \Sigma_r$ 
8 repeat
9   foreach  $A \rightarrow B \in \Sigma_{sr}$  do
10     foreach  $C \rightarrow D \in \Sigma_{sr}$  do
11       if  $A \subseteq C$  then
12         if  $C \subseteq A \cup B$  then
13           replace  $A \rightarrow B$  and  $C \rightarrow D$  by
14              $A \rightarrow BD$  in  $\Sigma_{sr}$ ;
15         else if  $D \subseteq B$  then
16           remove  $C \rightarrow D$  from  $\Sigma_{sr}$ 
17         else replace  $C \rightarrow D$  by
18            $C - B \rightarrow D - B$  in  $\Sigma_{sr}$ ;
19 until  $\Sigma_{sr}$  es un punto fijo;
20 /*Fase 3: Generación de  $\Sigma_{dsr}$  por Strong Simplification de  $\Sigma_{sr}^*$ /
21  $\Sigma_{dsr} = \Sigma_{sr}$ 
22 foreach  $A \rightarrow B \in \Sigma_{dsr}$  and  $C \rightarrow D \in \Sigma_{dsr}$  do
23   if  $B \cap C \neq \emptyset \neq D \setminus (A \cup B)$  then
24     add  $AC - B \rightarrow D - (AB)$  to  $\Sigma_{dsr}$ 
25 /*Fase 4: Generación de  $\Sigma_{do}$  por optimización de  $\Sigma_{dsr}^*$ /
26  $\Sigma_{do} = \emptyset$ 
27 foreach  $A \rightarrow B \in \Sigma_{dsr}$  do
28   foreach  $C \rightarrow D \in \Sigma_{dsr}$  DO
29     if  $C = A$  then  $B = B \cup D$ ;
30     if  $C \not\subseteq A$  then  $B = B \setminus D$ ;
31     if  $B \neq \emptyset$  then add  $A \rightarrow B$  to  $\Sigma_{do}$ ;
32 return  $\Sigma_{do}$ 
```

3.3. Implementaciones

En esta sección se describe la implementación de los algoritmos anteriores en Java y un ejemplo de uso. Ambos algoritmos implementan la interfaz `es.uma.pfc.is.algorithms.Algorithm` para que puedan ser ejecutadas por IS Bench. En la siguiente figura, se muestra el diagrama que muestra la jerarquía de clases implementada.

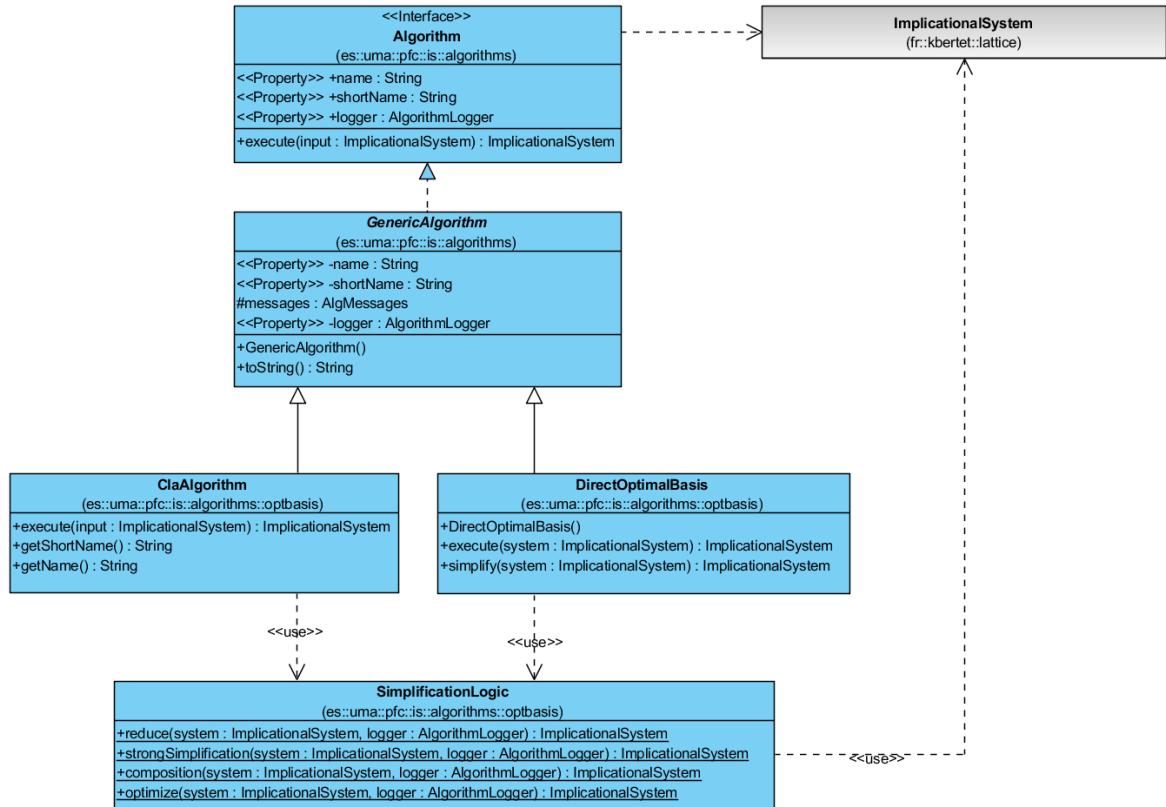


Figura 3.4: Implementación de Algoritmos

Los algoritmos tienen un parámetro de entrada de tipo `ImplicationalSystem`, que es el sistema implicacional inicial y devuelve un valor del mismo tipo con el sistema implicacional final. Ambos hacen uso de la clase `es.uma.pfc.is.algorithms.optbasis.SimplificationLogic` donde se implementan las reglas de equivalencia de la Lógica de Simplificación (Simplification Logic) mencionada en el artículo [1].

CLA

Clase: `es.uma.pfc.is.algorithms.optbasis.ClaAlgorithm`

```

public class ClaAlgorithm extends GenericAlgorithm {
    @Override
    public ImplicationalSystem execute(ImplicationalSystem input) {
        ImplicationalSystem directOptimalBasis = new ImplicationalSystem(input);
        directOptimalBasis = SimplificationLogic.reduce(
  
```

```

        directOptimalBasis, getLogger());
directOptimalBasis = SimplificationLogic.strongSimplification(
        directOptimalBasis, getLogger());
directOptimalBasis = SimplificationLogic.composition(
        directOptimalBasis, getLogger());
directOptimalBasis = SimplificationLogic.optimize(
        directOptimalBasis, getLogger());
return directOptimalBasis;
}

@Override
public String getShortName() {
    return "cla";
}

@Override
public String getName() {
    return "CLA";
}
}

```

Direct Optimal Basis

Clase: es.uma.pfc.is.algorithms.optbasis.DirectOptimalBasis

```

public class DirectOptimalBasis extends GenericAlgorithm {
    /**
     * Executes the Direct Optimal Basis algorithm.
     * @param system Input system.
     * @return Direct optimal basis.
    */
    @Override
    public ImplicationalSystem execute(ImplicationalSystem system) {
        ImplicationalSystem directOptimalBasis = new ImplicationalSystem(system);
        // Stage 1 : Generation of sigma-r by reduction of sigma
        directOptimalBasis = SimplificationLogic.reduce(
            directOptimalBasis, getLogger());
        // Stage 2: Generation of sigma-sr by simplification
        // (left+right) + composition of sigma-r
        directOptimalBasis = simplify(directOptimalBasis);
        // Stage 3: Generation of sigma-dsr by completion of sigma-sr
        directOptimalBasis = SimplificationLogic.strongSimplification(
            directOptimalBasis, getLogger());
        // Stage 4: Composition of sigma-dsr
    }
}

```

```
    SimplificationLogic.composition(directOptimalBasis, getLogger());
    // Stage 5: Generation of sigma-do by optimization of sigma-dsr
    directOptimalBasis = SimplificationLogic.optimize(
        directOptimalBasis, getLogger());
    return directOptimalBasis;
}

@Override
public String getShortName() {
    return "do";
}

@Override
public String getName() {
    return "Direct Optimal Basis";
}
}
```


Capítulo 4

HERRAMIENTA PARA IMPLICACIONES

El PFC *Algoritmos para la manipulación de implicaciones en ACF* tiene como objetivo la implementación de algoritmos relacionados con cierres y cálculo de bases y bases directas y realizar comparativas entre ellos.

Por esto, surge la idea de desarrollar una herramienta que permita ejecutar los algoritmos implementados, medir su rendimiento y guardar los resultados y estadísticas de éstos permitiendo compararlos entre sí. A esta herramienta se le ha llamado IS Bench. Además, debido a que no existe un benchmark (banco de pruebas) al estilo de los que se utilizan en lógica para la comparación de algoritmos, se hace necesario un generador de conjuntos de implicaciones aleatorios que pueda servir de entrada a los algoritmos en cuestión.

4.1. IS Bench

IS Bench es una aplicación cuyo objetivo es ejecutar algoritmos sobre sistemas implicacionales, guardar sus resultados y compararlos posteriormente. Esta herramienta se pretende ofrecer a la comunidad FCA, para incorporarla en un futuro cercano a la librería de la Dra. K.Bertet para comparativas de algoritmos relacionados con las bases.

La información se organiza en *workspaces*, que no son más que directorios que contendrán los algoritmos y benchmarks registrados y ejecutados en él, así como los resultados (sistemas de salida, trazas y estadísticas) generados.

Un *benchmark* o *experimento* es un conjunto de algoritmos que se ejecutan con uno o varios sistemas implicacionales de entrada comunes.

Un *algoritmo* es una implementación Java que recibe como entrada un sistema implicacional y devuelve otro equivalente. Los algoritmos implementados inicialmente son algoritmos de cálculo de bases directas-optimales, pero la API no restringe su uso a este tipo de bases ya que la única condición es que su entrada sea un sistema implicacional al igual que la salida.

El menú principal consta de las opciones:

- Preferencias → Workspaces, para la configuración de workspaces.
- Help con la ayuda de usuario.

El funcionamiento básico de la aplicación se resume en registro benchmarks, ejecución de éstos y visualización de resultados. Por esto la aplicación se divide en tres zonas principales: *Inicio*, *Benchmarks* y *Resultados*.

Con el área *Inicio* se pretende que el usuario pueda recordar las últimas acciones en el workspace actual al entrar en la aplicación.

Se muestra un resumen de las últimas ejecuciones que contiene el nombre del benchmark, la fecha y hora de la última ejecución y un enlace para ejecutarlo de nuevo. La pestaña *Benchmarks*, es el área en el que se definen y ejecutan los benchmarks. Se divide a su vez en dos pestañas: *Nuevo* y *Ejecutar*. Como sus nombres indican, en la pestaña *Nuevo* se pueden registrar nuevos benchmarks y en *Ejecutar*, ejecutarlos.

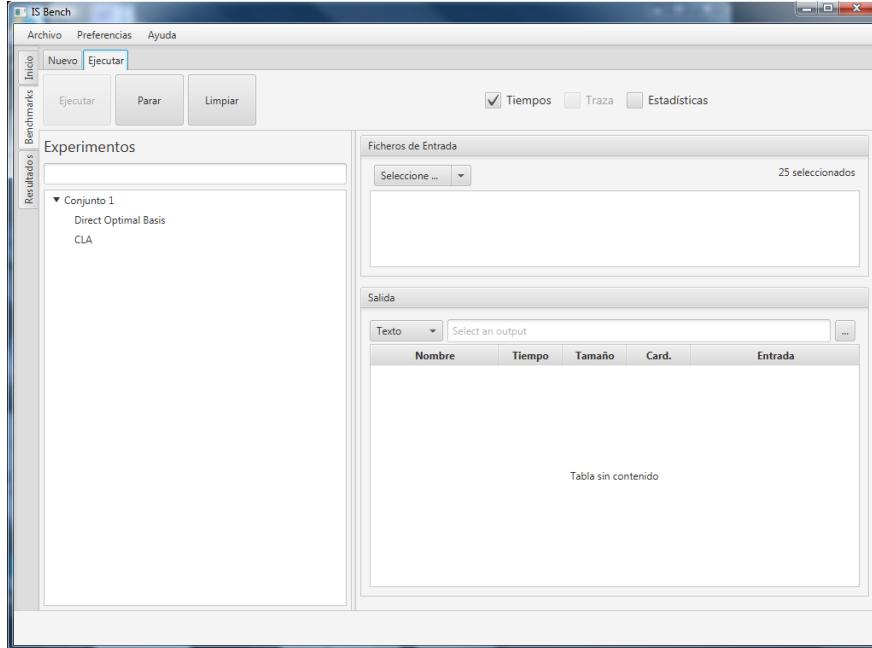


Figura 4.1: Pestaña Benchmarks

El área *Resultados*, contendrá los resultados detallados de los benchmarks del workspace actual, y será la zona que el usuario utilizará para realizar sus comparativas.

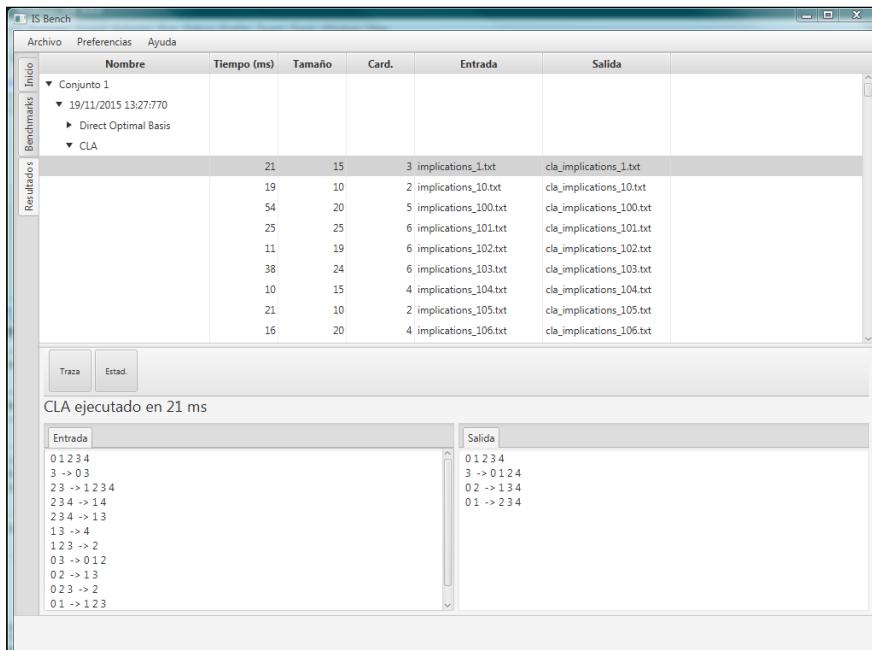


Figura 4.2: Pestaña Resultados

4.1.1. Workspaces

La configuración de la aplicación está organizada en *workspaces* que no son más que directorios en los que se guardan archivos relacionados para su uso en IS Bench. Estos archivos pueden ser:

- Registro de benchmarks.
- Archivos con sistemas implicacionales de entrada para los algoritmos a ejecutar.
- Archivos con los sistemas implicacionales de salida de los algoritmos ejecutados.
- Resultados.
- Librerías de algoritmos.

La primera vez que se arranca la aplicación, se crea el archivo *isbench.properties* en el directorio *isbench* del *home* del usuario. En Windows por ejemplo sería *C:\Users\usuario\isbench*.

Este archivo contiene las rutas de los workspaces registrados y el último workspace con el que el usuario entró a la aplicación. Por ejemplo:

```
default=C:\\\\Users\\\\Usuario\\\\isbench\\\\default
workspace.current=default
```

Figura 4.3: *isbench.properties*

Como se ve en la Figura 4.4 la carpeta de un workspace contiene los benchmarks registrados en él, la carpeta *lib* donde se pueden añadir librerías con implementaciones de algoritmos externas y un archivo de preferencias. También contiene un directorio por benchmark registrado que almacenará las entradas y salidas de sus ejecuciones.

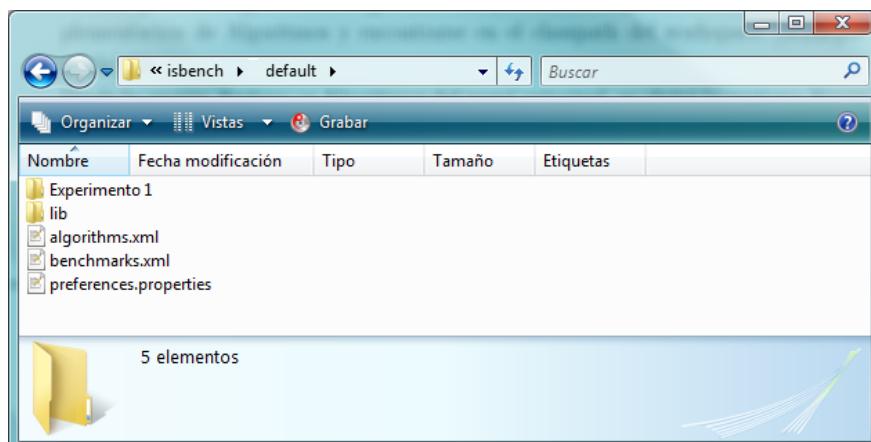


Figura 4.4: Contenido de un Workspace

La primera vez que el usuario arranca la aplicación, se establece un workspace por defecto en [home_usuario]\isbench\default. Desde la ventana *Workspaces*, que se puede acceder desde la opción de menú *Preferencias → Workspaces*, el usuario puede:

- Consultar el workspace actual y su configuración.
- Cambiar de workspace.
- Crear un nuevo workspace.

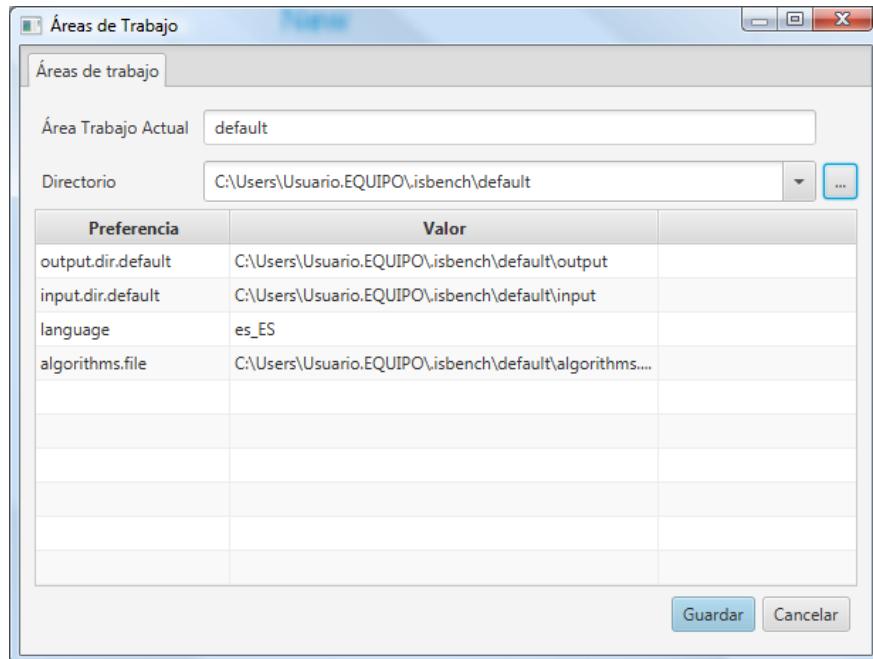


Figura 4.5: Ventana Áreas de Trabajo

El campo *Directorio* es un desplegable editable, que contiene como ítems los workspaces existentes.

Para cambiar de workspace, el usuario sólo tiene que seleccionar uno de los ítems del desplegable.

Para crear uno nuevo, sólo ha de introducir la ruta absoluta del directorio en el que quiere crearlo o seleccionarlo con el buscador pulsando en el botón .

4.1.2. Registrar Benchmarks

Un benchmark es la agrupación de un conjunto de algoritmos que se ejecutan con una misma entrada para la posterior comparación de sus resultados.

Para poder ejecutar un benchmark éste ha debido ser registrado en el workspace y esto se puede hacer desde la pestaña *Nuevo* del área *Benchmarks*.

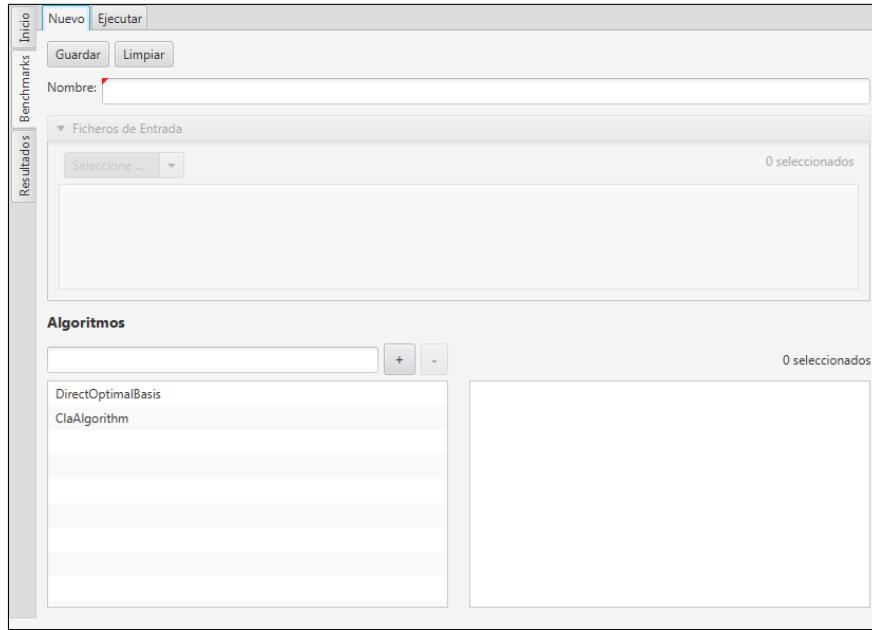


Figura 4.6: Pestaña Nuevo Benchmark

En ésta se muestra un formulario con los campos que se ven en la imagen:

- **Nombre:** Obligatorio y único. Nombre identificativo del benchmark.
Es el nombre que se muestra cuando se carga en listas o tablas.
 - **Ficheros de Entrada:** Lista de archivos que contienen los sistemas implicacionales de entrada.
Este campo se puede introducir manualmente, seleccionándolo mediante el explorador (opción *File* del botón ...) o creándolo con el generador de implicaciones (opción *Random* del botón ...).
 - **Algoritmos:** Obligatorio. Hay dos listas. La de la izquierda contiene los algoritmos encontrados en la carpeta *lib* del workspace y los algoritmos implementados en este proyecto, CLA y Direct Optimal Basis que se incluyen por defecto. En la de la derecha se irán añadiendo los algoritmos seleccionados con doble click en la primera lista.
Para filtrar la lista de algoritmos disponibles se puede hacer uso del filtro situado en la parte superior de la primera lista.
- + -
- Con los botones + y -, se pueden añadir y quitar librerías de algoritmos al workspace. Estos algoritmos deben cumplir las especificaciones que se indican en la sección 4.4.

Si se pulsa el botón *Guardar* con algún campo obligatorio sin completar, la aplicación

muestra un mensaje informando de ello y los campos *Nombre* y *Entrada* se marcan como erróneos.

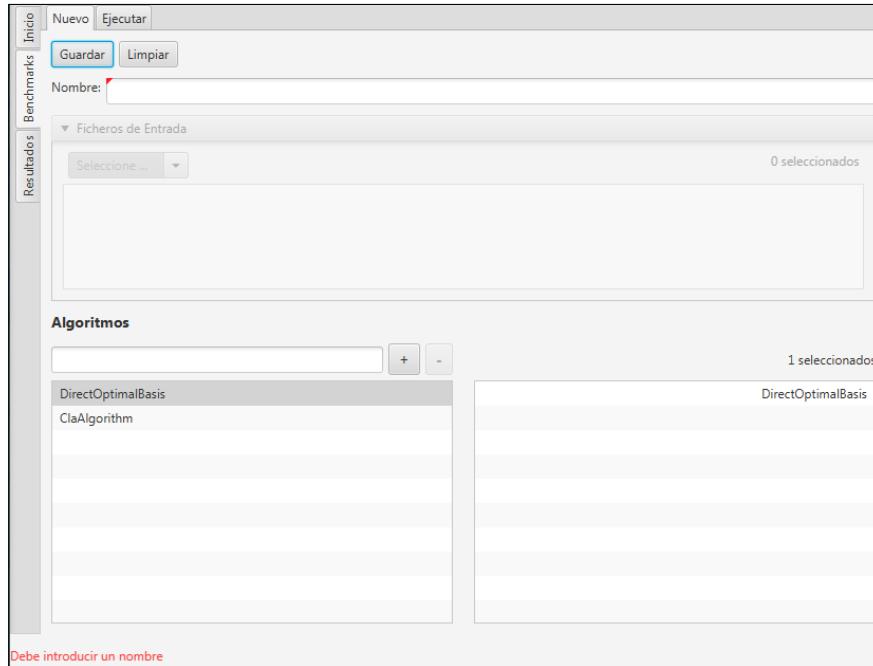


Figura 4.7: Error de validación del campo Nombre

Una vez introducidos todos los valores correctamente, al pulsar el botón *Guardar*, el benchmark se registra en el archivo *[workspace_dir]/benchmarks.xml*:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<benchmarks>
  <benchmark name="Experimento 1" workspace="C:\Users\Usuario.EQUIPO\.isbench\default">
    <inputsDir>C:\Users\Usuario.EQUIPO\.isbench\default\Experimento 1\input</inputsDir>
    <algorithms>
      <algorithm>
        <name>Direct Optimal Basis</name>
        <shortName>do</shortName>
        <type>es.uma.pfc.is.algorithms.optbasis.DirectOptimalBasis</type>
      </algorithm>
    </algorithms>
  </benchmark>
</benchmarks>
```

Figura 4.8: benchmarks.xml

4.1.3. Ejecutar algoritmos: Modos de ejecución

Recordemos que el objetivo de la ejecución de un algoritmo es, además de obtener un sistema implicacional de salida, obtener trazas y estadísticas que proporcionen información adicional para su comparativa.

Para lo que se distinguen tres modos de ejecución:

- **Tiempos:** Se mide el tiempo de ejecución del algoritmo y se registra.

- **Con traza:** En este modo, se genera el archivo *[nombre_archivo_salida]_history.log*, que contiene la traza de la ejecución.

Esta traza será la que el desarrollador, en el momento de la implementación del algoritmo, considere que pueda ser de interés para el investigador.

[nombre_archivo_salida] es el nombre base del archivo seleccionado para la salida del algoritmo. P.e., si el archivo que se ha tomado como salida es *do_output.txt*, el archivo de traza será *do_output_history.log*.

- **Con Estadísticas:** En este modo, se genera el archivo *[nombre_archivo_salida].csv*, en el que se guarda la evolución de los tamaños del sistema implicacional procesado. Esto se hace al final de la ejecución de cada algoritmo y se miden el tamaño y la cardinalidad del sistema obtenido.

[nombre_archivo_salida] es el nombre base del archivo seleccionado para la salida del algoritmo. Por ejemplo, si el archivo que se ha tomado como salida es *do_output.txt*, el archivo con las estadísticas será *do_output.csv*.

La información se guarda en archivos .csv para facilitar su visualización mediante tablas y gráficos.

Estos modos no son excluyentes y pueden combinarse como el usuario considere necesario.

4.1.4. Ejecutar Benchmarks

La ejecución de un benchmark consiste en la ejecución secuencial del conjunto de algoritmos que lo componen con uno o varios sistemas implicacionales de entrada en común.

Se genera un archivo de salida y varios adicionales según el modo de ejecución, por algoritmo ejecutado, además de un resumen de dicha ejecución con:

- La fecha y hora de ejecución
- Algoritmo ejecutado
- Sistema implicacional de entrada
- Sistema implicacional de salida
- Rutas de trazas y estadísticas generadas

Esta información se usará para la consulta de resultados.

La ejecución de benchmarks se hace desde la pestaña *Ejecutar* del área *Benchmarks*.

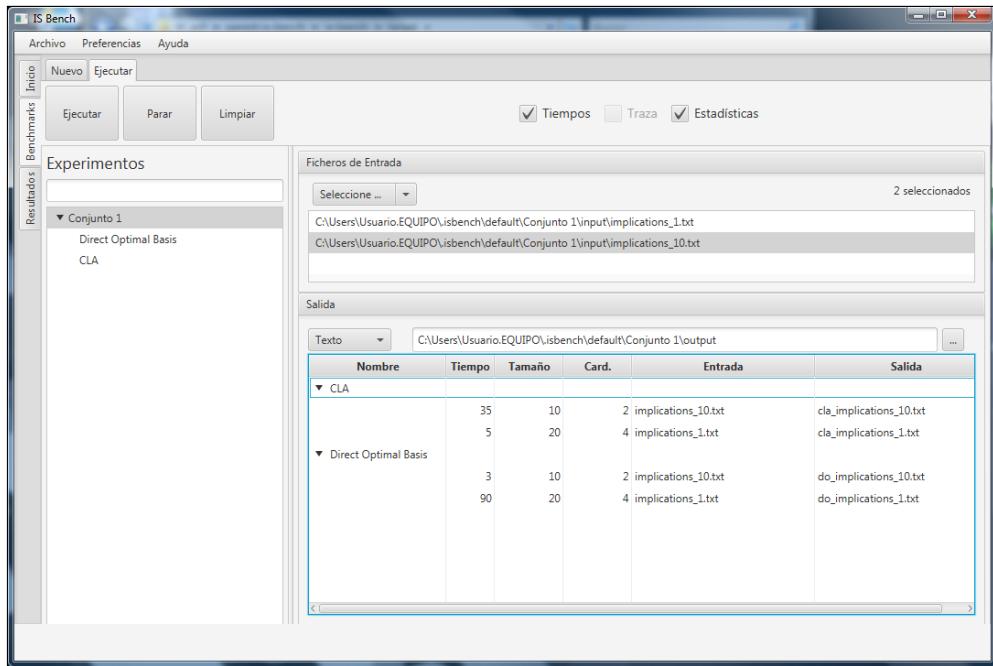


Figura 4.9: Pestaña Ejecutar Benchmark

Como se ve en la imagen, esta pestaña se compone de:

- Barra de herramientas
- Listado de Benchmarks
- Formulario para introducción de parámetros
- Visor de resultados

La barra de herramientas contiene tres botones *Ejecutar*, que ejecuta un benchmark o algoritmo seleccionado, *Parar* que para la ejecución en curso y *Limpiar* que limpia tanto la selección en el árbol como el formulario.

En la zona izquierda, se encuentra un árbol de dos niveles que contiene los benchmarks registrados en el workspace actual en el primer nivel, y los algoritmos que forman el benchmark en el segundo nivel. El contenido de éste árbol puede ser filtrado con el filtro situado en la parte superior de éste.

El formulario de introducción de parámetros consta de:

- **Modo de ejecución:** Tres modos posibles *Tiempos*, *Traza* y *Estadísticas* no excluyentes. Si se selecciona un benchmark en el árbol, el modo *Traza* se deshabilita.

Si se selecciona un algoritmo, los tres modos de ejecución están disponibles.

- **Ficheros de entrada:** Lista de los ficheros que contienen los sistemas implicacionales que servirán como entrada a los algoritmos a ejecutar. Para seleccionarlos, en el botón *Seleccionar Entrada* se presentan dos opciones: *Fichero* para seleccionar un fichero existente y *Aleatorio* para generar un sistema implicacional aleatorio con el Generador de Implicaciones.
- **Salida:** Directorio en el que se guardarán los archivos resultantes de la ejecución. La ruta se puede introducir manualmente o seleccionándolo pulsando el botón . La salida se puede generar en dos formatos: texto y prolog, que se pueden seleccionar en el desplegable que se encuentra a la izquierda del campo que contiene la ruta. El formato texto es la utilizada por defecto en la librería *javalattice*.

Los archivos generados tomarán nombres por defecto:

- de salida: [abreviatura_algoritmo]_[nombre_archivo_entrada].txt
- traza: [abreviatura_algoritmo]_trace.log
- estadísticas: [nombre_algoritmo]_[fecha_hora_ejecucion].csv, donde [fecha_hora_ejecucion] es el momento de la ejecución con formato yyyy-mm-dd hhMMss.

Bajo el campo *Salida*, se encuentra la tabla de resultados que contendrá los resultados de la ejecución actual agrupados por algoritmo. El contenido de cada una de las columnas es:

- **Nombre:** Sólo tiene valor en el primer nivel, y contiene el nombre del algoritmo.
- **Tiempo:** Tiempo de ejecución.
- **Tamaño:** Tamaño del sistema implicacional de salida.
- **Card.:** Cardinalidad del sistema implicacional de salida.
- **Entrada:** Nombre del archivo que contiene el sistema implicacional de entrada.
- **Entrada:** Nombre del archivo que contiene el sistema implicacional de salida.

Al hacer click con el botón derecho sobre alguno de los resultados, se muestra un menú contextual con las siguientes opciones:

- **Mostrar Log:** Muestra la traza generada si se ha ejecutado con el modo *Tiempos* activo como se ve en la Figura 4.10. También se podrá ver dicha traza haciendo doble click sobre el resultado.

- **Fichero de Estadísticas:** Abre el archivo de estadísticas generado si se ha ejecutado con el modo *Estadísticas* activo.

```

C:\Users\Usuario\isbench\default\Optimal Basis\output\20151001192050\do_trace.log

*****
INPUT:
012345
3 -> 4
34 -> 05
34 -> 014
24 -> 1245
245 -> 234
235 -> 012
145 -> 012
1345 -> 012
1234 -> 034
05 -> 234
02 -> 0345
023 -> 1245
01234 -> 0234

*****
GeneraciOn de IS reducido
*****


012345
3 -> 4
34 -> 05
34 -> 01
24 -> 15
245 -> 3
235 -> 01
145 -> 02
1345 -> 02
1234 -> 0

```

Figura 4.10: Visor de traza

4.1.5. Consulta de Resultados

En el área de *Resultados*, se pueden consultar los resultados de las ejecuciones realizadas en el workspace actual. Éstos se presentan en una tabla en forma de árbol agrupados por benchmark, fecha y hora de ejecución y algoritmo. En el último nivel, se muestra una fila por entrada procesada con el tiempo de ejecución, el archivo de entrada y el archivo de salida como se ve en la siguiente imagen.

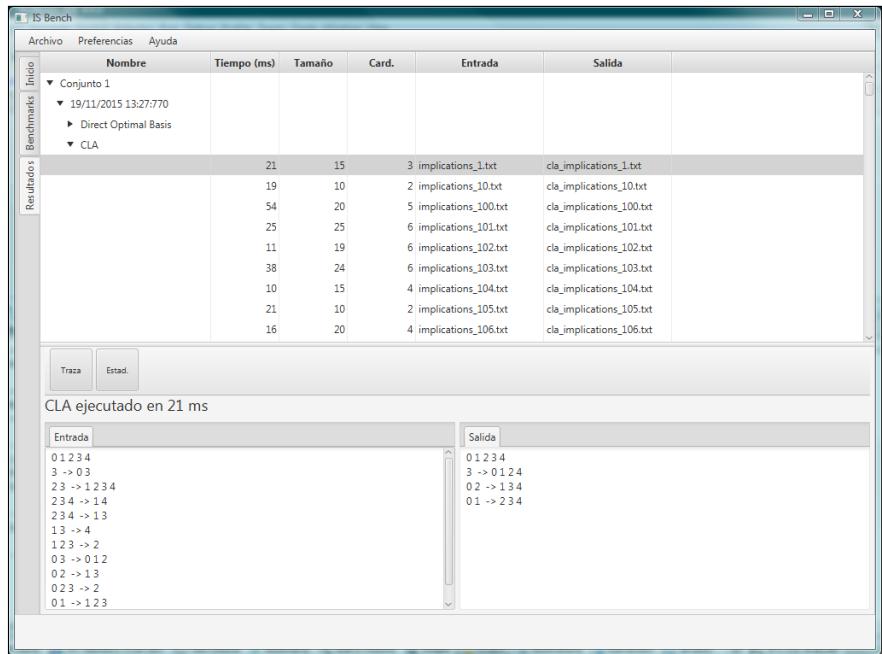


Figura 4.11: Resultados

Si se selecciona una de éstas filas, se muestra en la parte inferior sistemas implicacionales de entrada y salida. Además se puede consultar la traza generada y el archivo de estadísticas mediante los botones *Traza* y *Estadísticas*.

4.2. Generador de sistemas implicacionales aleatorios

El generador de sistemas implicacionales, permite a partir de un número de argumentos y un número de implicaciones, generar un sistema implicacional aleatorio. Se utiliza la librería *java-lattices* de la doctora Karelle Bertet, para el cálculo del sistema a partir de estos parámetros.

Como se ve en la Figura 4.12, este generador es una herramienta sencilla que consta de una pantalla principal dividida en tres zonas bien diferenciadas:

- Barra de herramientas
- Panel izquierdo en el que se introducen los parámetros para la creación del sistema: número de atributos, número de implicaciones, tipo, máximo y mínimo número de atributos en premisas y conclusiones y número de sistemas.
- Panel derecho que contiene el campo *Salida* con la ruta absoluta del fichero en el que se guarda el sistema implicacional generado, y un visor en el que se muestra dicho sistema.

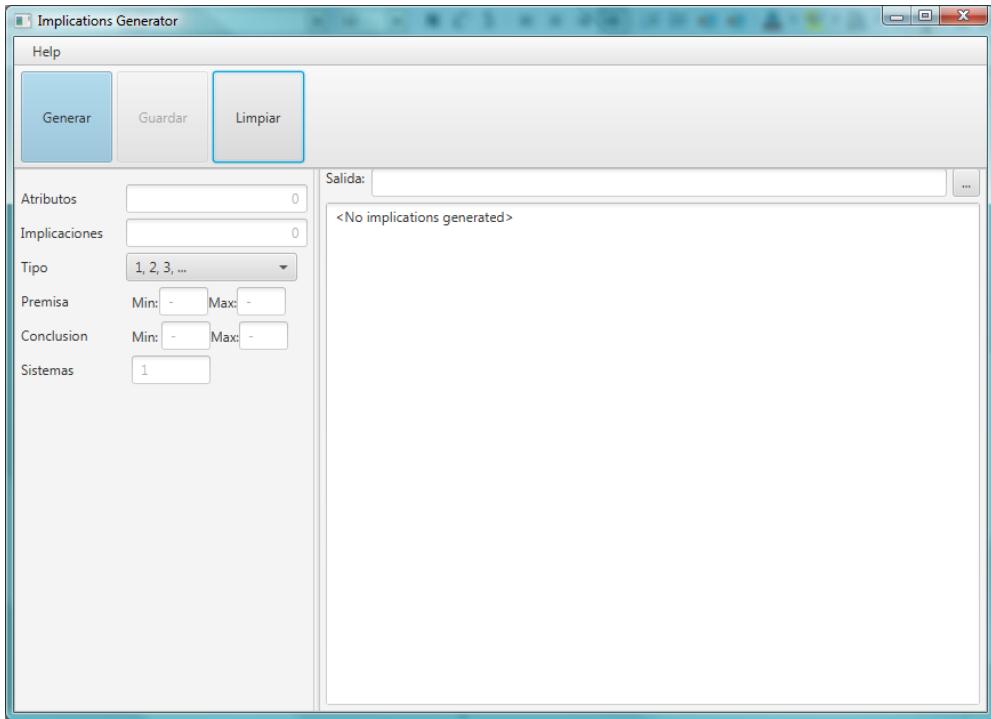


Figura 4.12: Generador de Implicaciones

4.2.1. Generar un sistema implicacional

Para generar un sistema implicacional se deben seguir los siguientes pasos:

1. Introducir los parámetros para el sistema.
 - **Atributos:** Número de atributos.
 - **Implicaciones:** Número de implicaciones.
 - **Tipo:** Tipo de los atributos. Se puede seleccionar entre tres tipos: numérico (1, 2, 3,...) , alfabético (a, b, c, ...), alfanumérico (a1, a2, a3, ...).
 - **Premisa:** Número mínimo y máximo de atributos en la premisa.
 - **Conclusion:** Número mínimo y máximo de atributos en la conclusión.
 - **Sistemas:** Número de sistemas implicacionales a generar. Por defecto, 1.
2. Pulsar el botón *Generar*. Al pulsar este botón se genera el sistema con los parámetros introducidos y se muestra en el visor.

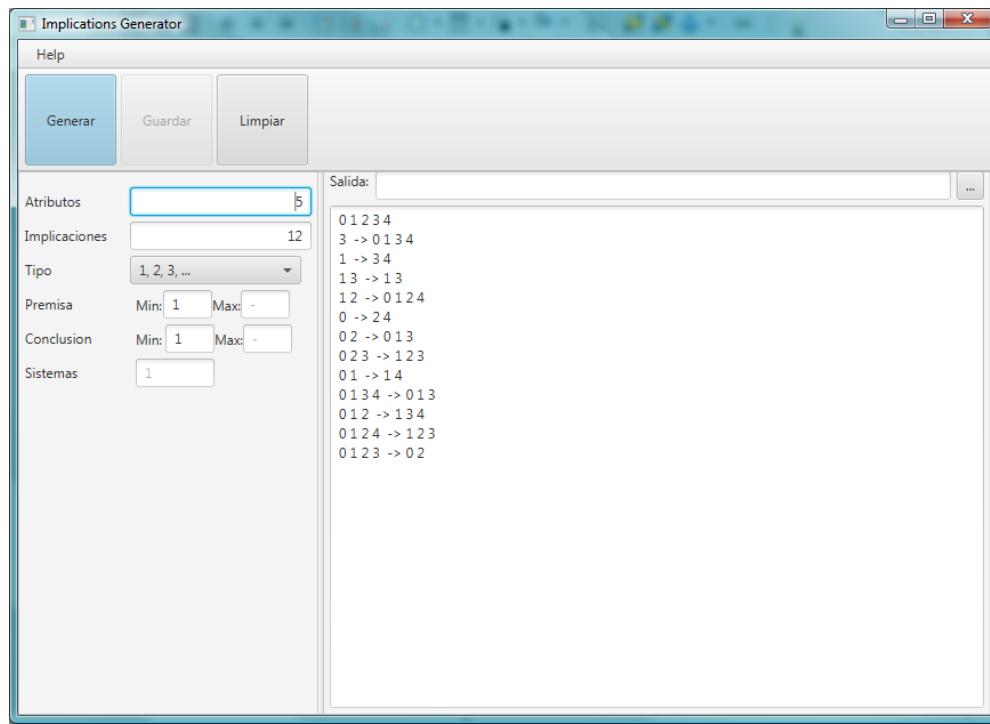


Figura 4.13: Generación de sistema aleatorio

3. Guardar el sistema en un fichero si se desea.

Para guardar el sistema, se debe introducir previamente la ruta absoluta del fichero en el que se desea guardar. Se puede hacer introduciéndolo directamente en el campo de texto, o buscándolo en el explorador, haciendo click en el botón *Examinar* () del campo *Salida*.

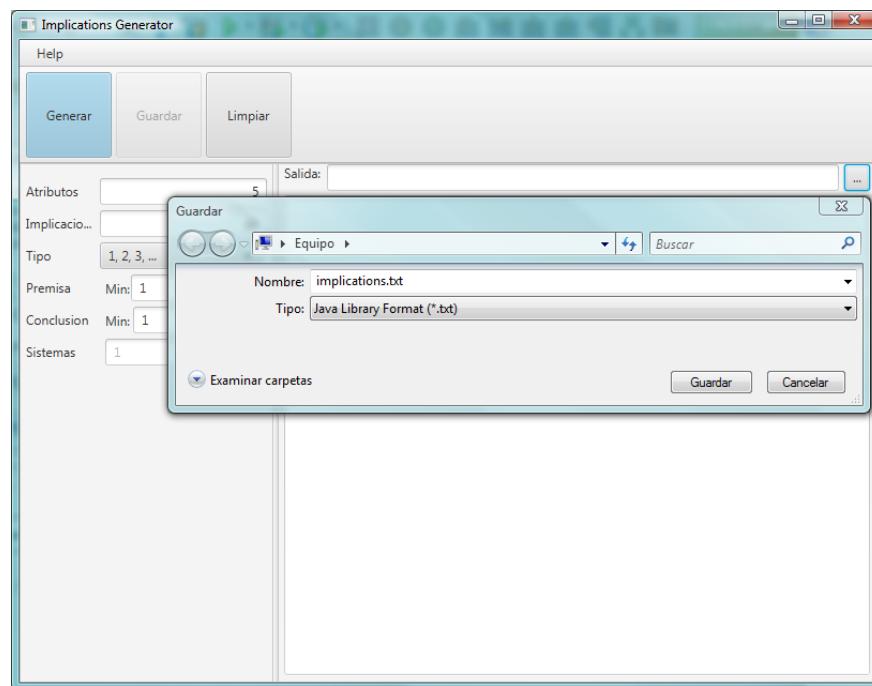


Figura 4.14: Selección de archivo de salida

El sistema se puede guardar en dos formatos: archivo de texto (txt) y archivo Prolog (pl).

Una vez introducida la ruta del archivo, el botón *Guardar* se habilitará y al pulsar sobre él se creará o actualizará el archivo con el sistema implicacional generado.

4.2.2. Generar n sistemas implicacionales

Como se ha comentado anteriormente, es posible generar n sistemas implicacionales con los mismos parámetros. Para ello, sólo se ha de introducir en el campo *Sistemas* el número de conjuntos de implicaciones a generar. En este modo de ejecución, los conjuntos de implicaciones generados no se previsualizan y se guardan directamente en fichero. Por ello, es obligatorio que el campo *Salida* contenga la ruta absoluta del fichero base, de otra forma, el botón *Generar* no se habilitará.

A partir de esta ruta, se guardarán los n conjuntos de implicaciones generados en n archivos, tomando su nombre a partir del introducido en el campo *Salida* y añadiéndole un índice.

Por ejemplo, si el campo *Salida* contiene la ruta C:\implications\system.txt y se van a generar tres conjuntos de implicaciones, se crearán los archivos:

- C:\implications\system_0.txt
- C:\implications\system_1.txt
- C:\implications\system_2.txt

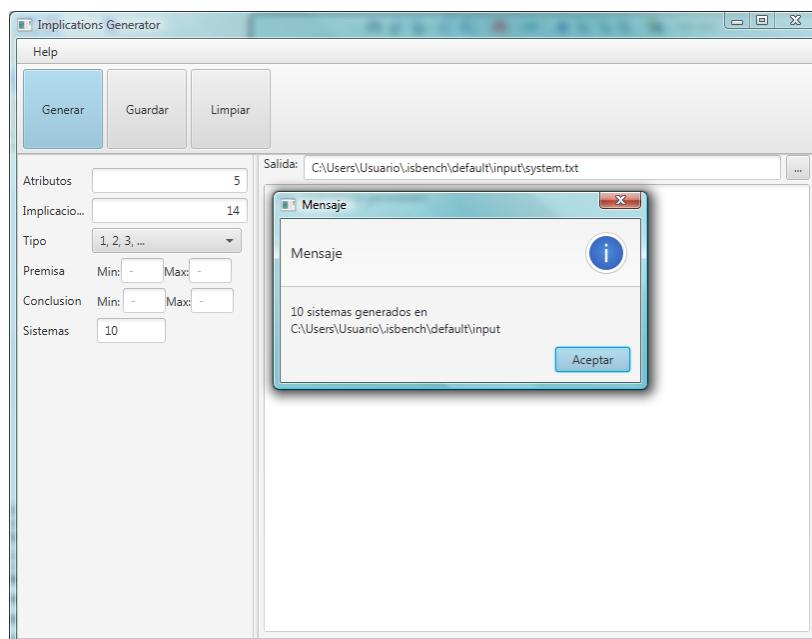


Figura 4.15: Generar n sistemas implicacionales

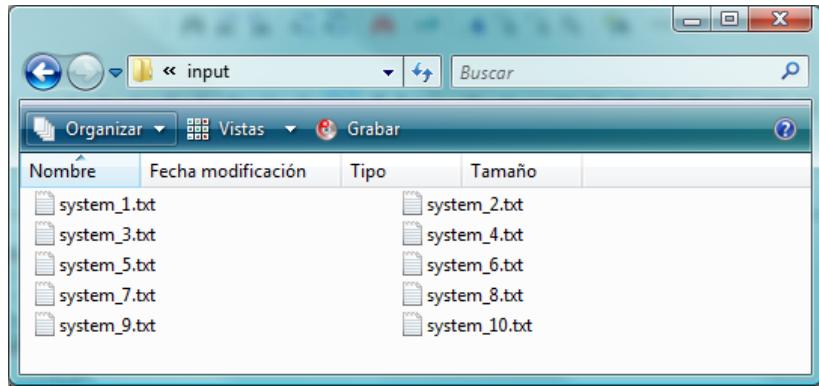


Figura 4.16: Sistemas implicacionales generados

Desde IS Bench se puede acceder al Generador de Implicaciones, para seleccionar las entradas de los benchmarks y algoritmos. Se puede hacer en:

1. el registro de Benchmarks (*Benchmarks → Nuevo*).



Figura 4.17: IS aleatorio para el registro de benchmarks

2. la ejecución de benchmarks o algoritmos (*Benchmarks → Ejecutar*).

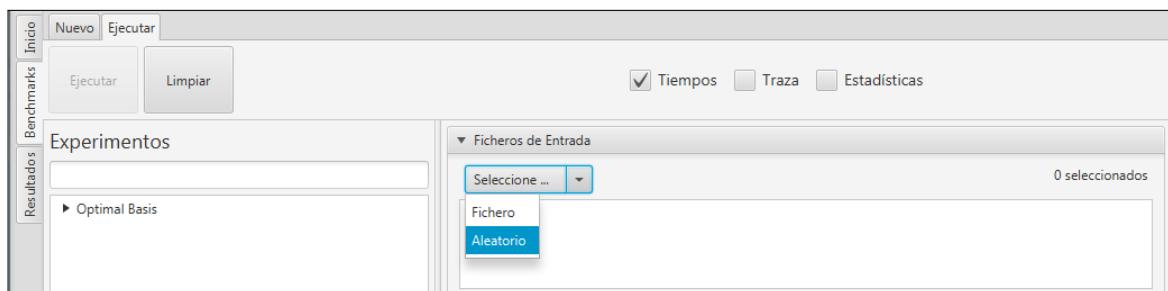


Figura 4.18: IS aleatorio para la ejecución de benchmarks o algoritmos

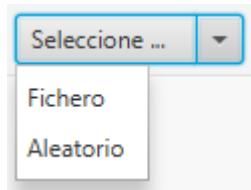
4.3. Ejemplo de uso

En esta sección se muestra un ejemplo de uso de la herramienta, desde que se registra el benchmark hasta la consulta de resultados, pasando por su ejecución. El ejemplo

será de un benchmark en el que se compararán los algoritmos CLA y Direct Optimal Basis con un conjunto de implicaciones aleatorias.

1. Registrar un nuevo benchmark en la pestaña *Nuevo* del área *Benchmarks*.

- a) Introducir el nombre en el campo *Nombre*.
- b) Seleccionar la opción *Aleatorio* del desplegable *Seleccione Entrada*.



- c) Se abre el generador de implicaciones. Introducir el número de atributos, implicaciones y conjuntos a generar en sus correspondientes campos y pulsar el botón *Generar*.
- d) Al finalizar la generación, se muestra un mensaje informando de ello. Acepar el mensaje y cerrar el generador. Las rutas de las entradas generadas se cargan en la lista de entradas.

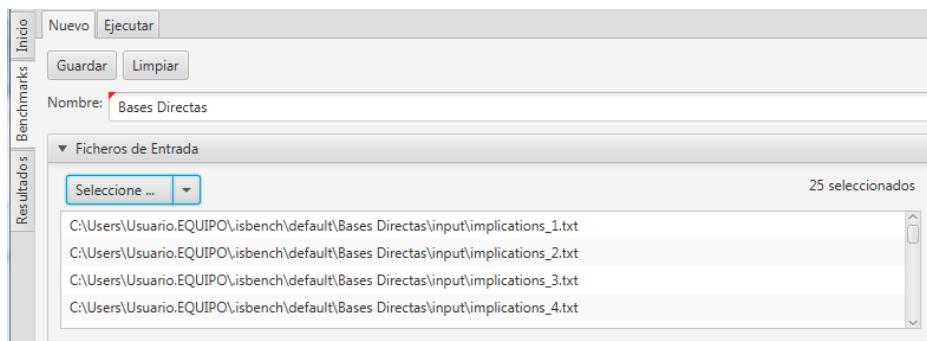


Figura 4.19: Entradas insertadas desde el Generador

- e) Seleccionar los algoritmos CLA y Direct Optimal Basis de la lista de algoritmos con doble click. Éstos se añadirán a la lista de la derecha.
- f) Hacer click en *Guardar*. Si todo ha ido bien, se mostrará un mensaje al pie informando de ello.

2. Ejecutar el benchmark creado desde la pestaña *Ejecutar*.

- a) El benchmark registrado aparece en el árbol de benchmarks a la izquierda. Seleccionarlo haciendo click sobre él.

- b) La lista *Ficheros de Entrada* se inicializa con las entradas definidas.
- c) El campo *Salida*, se inicializa con la ruta *[workspace_dir]/[nombre_benchmark]/output*. En este caso la ruta es un directorio, ya que se generarán n salidas, una o más por algoritmo ejecutado.
- d) El modo de ejecución por defecto es *Tiempos*. Seleccionar además el modo *Estadísticas* para que calcule los tamaños y cardinalidades de los sistemas implicacionales de salida.
- e) Pulsar el botón *Run*. Aparece un indicador que estará visible durante la ejecución.

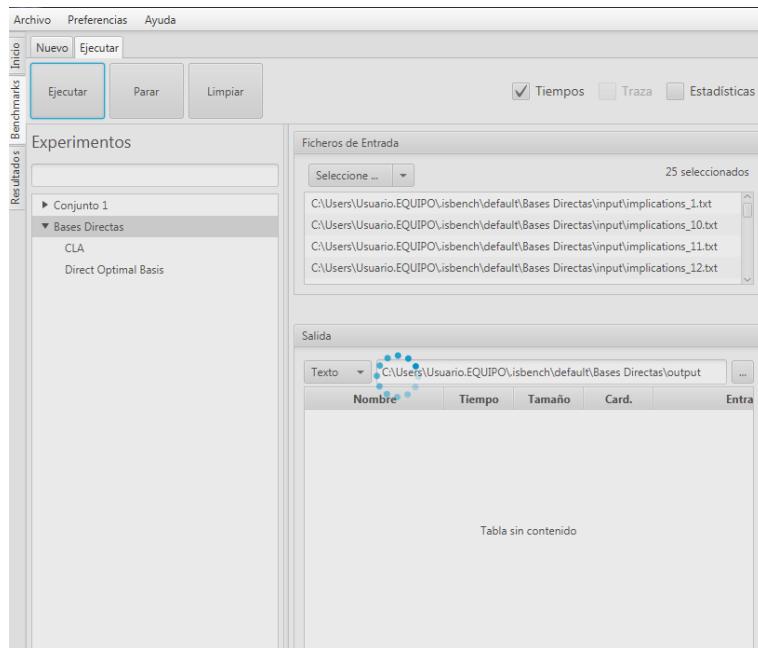


Figura 4.20: Ejecución de un benchmark

- f) Cuando finaliza la ejecución, el indicador desaparece y los resultados se muestran en la tabla de resultados.

Salida					
Nombre	Tiempo	Tamaño	Card.	Entrada	Salida
► Direct Optimal Basis					
▼ CLA					
94,237	15	3	implications_1.txt		cla_imPLICATIONS_1.txt
101,402	25	5	implications_10.txt		cla_imPLICATIONS_10.txt
98,975	15	3	implications_11.txt		cla_imPLICATIONS_11.txt
128,701	20	4	implications_12.txt		cla_imPLICATIONS_12.txt
24,095	10	2	implications_13.txt		cla_imPLICATIONS_13.txt
114,197	15	3	implications_14.txt		cla_imPLICATIONS_14.txt
207,472	15	3	implications_15.txt		cla_imPLICATIONS_15.txt
101,444	20	4	implications_16.txt		cla_imPLICATIONS_16.txt
262,652	20	4	implications_17.txt		cla_imPLICATIONS_17.txt

Figura 4.21: Resultados de la ejecución

4.4. Implementación de Algoritmos

Para que terceros puedan implementar algoritmos que puedan ser ejecutados por IS Bench, se ha creado una API con las interfaces y clases básicas para este fin que se incluye en el la librería *is-algorithms*.

El requisito mínimo para que un algoritmo pueda ser ejecutado desde IS Bench, es que implemente la interfaz `es.uma.pfc.is.algorithms.Algorithm` incluida en la librería **is-algorithms-1.0.0.jar**.

La dependencia a incluir en proyectos Maven es:

```
<dependency>
    <groupId>es.uma.pfc</groupId>
    <artifactId>is-algorithms</artifactId>
    <version>1.0.0</version>
</dependency>
```

La aplicación ejecuta el método

`execute(input : ImplicationalSystem) : ImplicationalSystem` por lo que éste será en el que se implemente el algoritmo en cuestión.

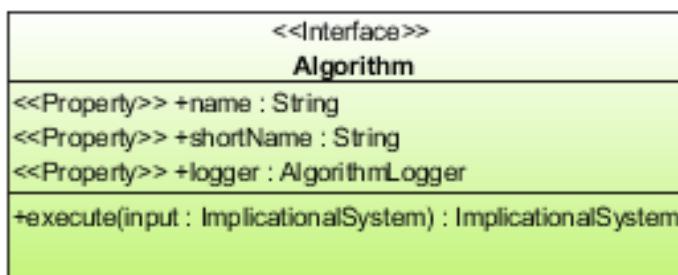


Figura 4.22: Interfaz Algorithm

La API proporciona una implementación abstracta de esta interfaz, `es.uma.pfc.is.algorithms.GenericAlgorithm`, que implementa los *getters* y *setters* obligatorios, así como utilidades para la sustitución, adición y eliminación de implicaciones del sistema. El método `execute(input : ImplicationalSystem) : ImplicationalSystem` no se implementa, ya que esto se deberá hacer en la implementación final.

GenericAlgorithm
<<Property>> -name : String
<<Property>> -shortName : String
#messages : AlgMessages
+GenericAlgorithm()
+getLogger() : AlgorithmLogger
#setLogger(logger : AlgorithmLogger) : void
#removeRule(system : ImplicationalSystem, rule : Rule) : void
#addRule(system : ImplicationalSystem, rule : Rule) : void
#addRuleAndElements(system : ImplicationalSystem, rule : Rule) : ImplicationalSystem
#addRuleAndElements(system : ImplicationalSystem, rule : Rule, trace : boolean) : ImplicationalSystem
#replaceRule(system : ImplicationalSystem, rule1 : Rule, rule2 : Rule) : void
#history(message : String, args : Object ...) : void
+toString() : String

Figura 4.23: Clase abstracta GenericAlgorithm

En el apartado 6.5.8 se incluyen los detalles de esta API.

A continuación se muestra un ejemplo de un algoritmo, que devuelve un sistema equivalente al de entrada cuyas conclusiones sólo contienen un atributo.

```
public class UnaryAlgorithm extends GenericAlgorithm {
    @Override
    public ImplicationalSystem execute(ImplicationalSystem system) {
        ImplicationalSystem outputSystem = new ImplicationalSystem(system);
        outputSystem.makeUnary();
        return outputSystem;
    }
}
```

Por lo que, para implementar un algoritmo y ejecutarlo desde IS Bench:

1. Crear un proyecto Java (JDK 1.8_u65+) que tenga como dependencia la librería *is-algorithms*.
2. Crear una clase que implemente la interfaz `es.uma.pfc.is.algorithms.Algorithm` o extienda de `es.uma.pfc.is.algorithms.GenericAlgorithm`.
3. En el método `execute(input : ImplicationalSystem) : ImplicationalSystem` implementar el algoritmo en cuestión.
4. Compilar el proyecto y generar el JAR correspondiente.
5. Añadir la librería al workspace desde la pestaña *Nuevo* del área *Benchmarks*, con el botón .

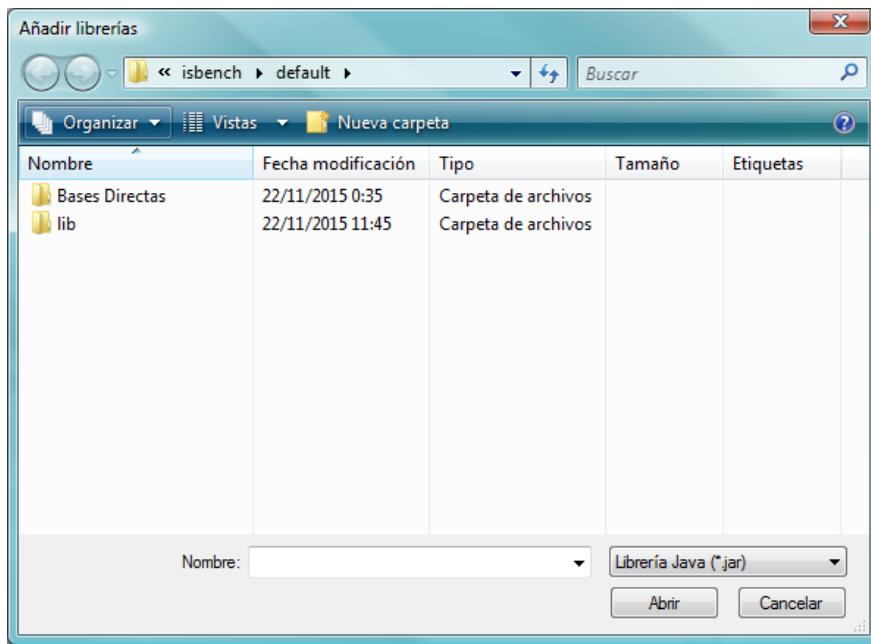


Figura 4.24: Añadir librerías

6. Incluir el algoritmo en un *benchmark* como se explica en el apartado 4.1.2, si no se ha hecho ya en el punto anterior.
7. Una vez incluido en el Benchmark, se podrá ejecutar el algoritmo bien individualmente o bien a través del Benchmark desde la pestaña *Benchmarks / Ejecución*, como se explica en 4.1.4.

Capítulo 5

COMPARATIVA ENTRE ALGORITMOS

En este capítulo, se realiza una comparativa entre los algoritmos implementados (Direct Optimal Basis y CLA) obteniendo los resultados con IS Bench.

El siguiente experimento está basado en conjuntos de implicaciones aleatorias. Se han generado 50 conjuntos combinando dos parámetros: el número de implicaciones (de 5 a 15) y el número de atributos (de 5 a 15).

Se han desarrollado dos clases para la comparación de los dos algoritmos cuyos detalles se han explicado en el capítulo 3 *Algoritmos de cálculo de bases*. El experimento ha sido ejecutado en un Intel Core i5, 8GB de RAM, con Windows 7 64 bits.

En la Figura 5.1 se muestran para cada par número de implicaciones atributos, los promedios de los tamaños, cardinalidades y tiempo de ejecución en segundos para ambos algoritmos.

[Implicaciones, Atributos]	Tamaño	Cardinalidad	Tiempo(seg.)	Tiempo(seg.)
			CLA	DO Basis
[5,5]	11,80	2,86	0,0017	0,0009
[5, 10]	16,64	3,76	0,0109	0,0039
[5, 15]	17,22	3,82	0,0162	0,0030
[10, 5]	35,80	4,82	0,1193	0,0921
[10, 10]	54,16	7,45	73,1097	41,5137
[10, 15]	70,48	9,00	224,7242	36,0385
[15, 5]	68,74	6,68	28,3056	8,5009

Figura 5.1: Resultados del experimento

En la Figura 5.2 se muestra una gráfica donde se puede apreciar que el tiempo de ejecución del algoritmo CLA aumenta a medida que aumenta la cardinalidad de los resultados. Sin embargo, el algoritmo Direct Optimal Basis también lo hace pero de una forma más moderada.

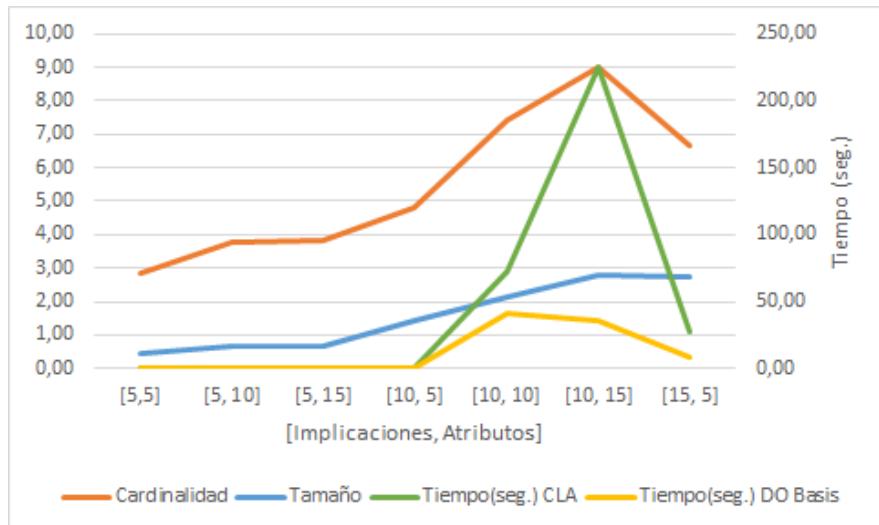


Figura 5.2: Gráfica de resultados

Capítulo 6

DISEÑO E IMPLEMENTACIÓN DE LA APLICACIÓN

6.1. Tecnologías y herramientas empleadas

En esta sección se enumeran las tecnologías y herramientas usadas a lo largo del proyecto.

- Análisis y modelado de la aplicación
 - Visual Paradigm 12.2
- Lenguaje de programación y frameworks
 - Ya que uno de los requisitos principales del proyecto es el uso del librería de la doctora K.Bertet implementada en Java, tanto las herramientas como los algoritmos, se han implementado en este lenguaje de programación en su versión 8
 - JavaFX 2 como framework para la interfaz de usuario.
 - JAXB 2.0 para la persistencia de información en archivos como registro de benchmarks, algoritmos y resultados.
 - JUnit 4.10 para tests unitarios.
- Herramientas de desarrollo
 - Netbeans 8.0.2.
 - Maven 3.
 - JavaFX Scene Builder 2.0 para el diseño de la interfaz de usuario.
- Documentación
 - Esta memoria ha sido redactada con Latex y Microsoft Word 2013.

6.2. Casos de uso

En 1986, Ivar Jacobson, importante contribuyente al desarrollo de los modelos de UML y proceso unificado, creó el concepto de caso de uso.

Durante los años 1990 los casos de uso se convirtieron en una de las prácticas más comunes para la captura de requisitos funcionales, especialmente con el desarrollo del paradigma de la programación orientada a objetos, donde se originaron, si bien puede utilizarse con resultados igualmente satisfactorios con otros paradigmas de programación.

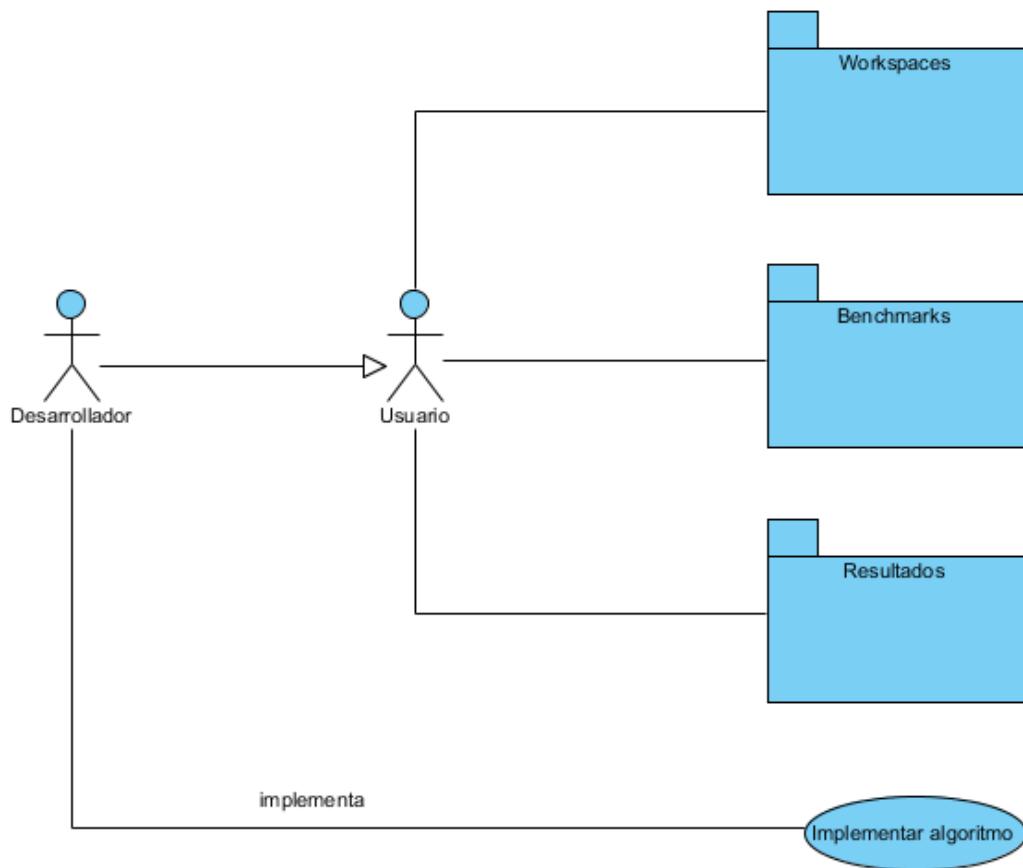
Un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio

sistema. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas.

La aplicación principal de los casos de uso es en el proceso de análisis y diseño pero de manera particular en la definición de requerimientos del usuario. Es una excelente herramienta de comunicación debido a la sencillez de su elaboración así como su comprensión.

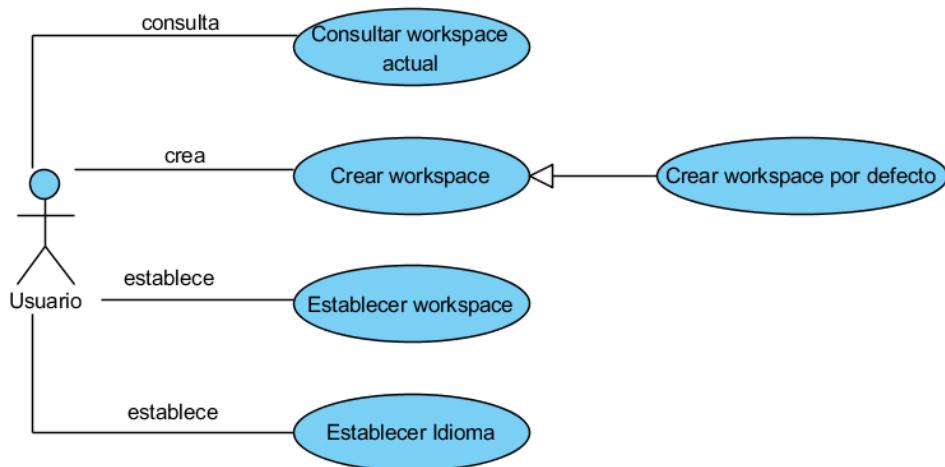
En esta sección, se presentan los diagramas de casos de uso diseñados para el análisis y diseño de la herramienta *IS Bench* y sus especificaciones.

6.2.1. Inicio



Name	Description
>User Desarrollador	Implementa los algoritmos que el sistema ejecuta.
User Usuario	Usuario general de la aplicación. No existirán roles ni permisos especiales. Cualquier usuario puede realizar cualquier acción.
Implementar algoritmo	Un desarrollador puede implementar algoritmos para que el sistema los ejecute. Éstos han de ser implementaciones Java que deben cumplir las siguientes condiciones: <ol style="list-style-type: none"> 1. Han de encontrarse en la ruta que se indique en la propiedad <i>algorithms.classpath</i> en las preferencias del <i>workspace</i> actual. Por defecto, es la carpeta <i>lib</i> de dicho <i>workspace</i>. 2. Debe implementar la interfaz <i>es.uma.pfc.is.algorithms.Algorithm</i>.
Workspaces	Gestión de workspaces.
Resultados	Consulta de resultados.
Benchmarks	Registro y ejecución de benchmarks y algoritmos.

6.2.2. Workspaces



Summary

Name	Description
Usuario	Usuario general de la aplicación.
Crear workspace por defecto	Creación del workspace inicial en el primer arranque de la aplicación.
Consultar workspace actual	Consultar información del workspace actual.
Establecer Idioma	<p>Es posible mostrar la aplicación en distintos idiomas, siendo así más amigable para el usuario.</p> <p>El idioma por defecto, será el establecido en la máquina.</p> <p>Para cambiar el idioma de la aplicación, se deberá acceder a la pestaña "Language" de la ventana "Preferences", que se puede abrir desde la opción de menú "Preferences".</p> <p>Los idiomas disponibles inicialmente son:</p> <ul style="list-style-type: none"> • Español • Inglés
Establecer workspace	Cambio de workspace.
Crear workspace	Crear un workspace.

Description

Para la comodidad de uso de la herramienta, se implementará un sistema de configuración de usuario de forma que éste pueda centralizar las entradas y salidas en un directorio de trabajo, dar la opción de recordar la última ubicación de la cual se ha seleccionado una entrada o una salida, organizar conjuntos de pruebas, etc.

Details

Usuario

Name	Value
Description	Usuario general de la aplicación. No existirán roles ni permisos especiales. Cualquier usuario puede realizar cualquier acción.
ID	AC01
Visibility	public

Crear workspace por defecto

Name	Value
Description	La primera vez que se ejecuta la aplicación, se crea automáticamente: <ul style="list-style-type: none">• El archivo de configuración general \.isbench\isbench.properties• El workspace por defecto con:<ul style="list-style-type: none">◦ El directorio \.isbench\default, \.isbench\default\input y \.isbench\default\output.◦ El archivo \.isbench\default\preferences.properties con las preferencias por defecto.• Se registra el workspace creado anteriormente en el archivo de configuración general \.isbench\isbench.properties con la propiedad: <code>workspace.default=\.isbench\default</code>• Se establece como workspace actual, el workspace por defecto estableciendo la propiedad workspace.current en el archivo El archivo de configuración general \.isbench\isbench.properties :

Name	Value
	workspace.current= workspace.default
ID	UC26
Stereotypes	UseCase
Requirements	Crear workspace por defecto

Crear workspace por defecto en la primera ejecución

1. Ejecutar por primera vez la aplicación.
2. **SYSTEM** Se crea los directorios \.isbench\default, \.isbench\default\input y \.isbench\default\output.
3. **SYSTEM** Se crea el archivo \.isbench\default\preference.properties.
4. **SYSTEM** El sistema crea el archivo \.isbench\isbench.properties.
5. **SYSTEM** Se establece como workspace actual el workspace creado, estableciendo la propiedad `workspace.current=\.isbench\default`

● Consultar workspace actual

Name	Value
Description	Consultar información del workspace actual.
ID	UC24
Stereotypes	UseCase
Requirements	Gestor de configuración general de la aplicación, Implementación de ventana "Workspaces"

Primer acceso al sistema

1. El usuario accede al sistema.
2. Selecciona la opción de menú "Preferencias -> Workspaces".
3. **SYSTEM** Se crea el archivo \.isbench\isbench.properties.
4. **SYSTEM** Se crea un workspace por defecto en el directorio \.isbench\default y lo registra en el archivo \.isbench\isbench.properties con la propiedad, `workspace.default=\.isbench\default`
5. **SYSTEM** Se establece como workspace actual el workspace creado, estableciendo la propiedad `workspace.current=\.isbench\default`
6. **SYSTEM** Se muestra la ventana "Workspaces" con el workspace actual y sus preferencias.

Consultar workspace actual

1. Acceder al menú "Preferencias -> Workspaces"
2. **SYSTEM** Se muestra la ventana "Workspaces" con el workspace seleccionado y las preferencias asociadas a éste.

Establecer Idioma

Name	Value
Description	<p>Es posible mostrar la aplicación en distintos idiomas, siendo así más amigable para el usuario.</p> <p>El idioma por defecto, será el establecido en la máquina.</p> <p>Para cambiar el idioma de la aplicación, se deberá acceder a la pestaña "Language" de la ventana "Preferences", que se puede abrir desde la opción de menú "Preferences".</p> <p>Los idiomas disponibles inicialmente son:</p> <ul style="list-style-type: none">• Español• Inglés
ID	UC13
Stereotypes	UseCase

Cambiar el idioma

1. Acceder a la opción de menú "Preferences"
2. Seleccionar la pestaña "Language"
3. Seleccionar un idioma del desplegable.
4. Pulsar el botón "Ok".

Cancelar el cambio de idioma

1. Acceder a la opción de menú "Preferences"
2. Seleccionar la pestaña "Language"
3. Seleccionar un idioma del desplegable.
4. Pulsar el botón "Cancel".

● Establecer workspace

Name	Value
Description	<p>El usuario podrá cambiar de workspace desde la ventana "Workspaces" que se abrirá desde la opción de menú "Preferences -> Workspaces".</p> <p>El usuario ha de pulsar el botón "Switch" y el sistema mostrará un desplegable con los workspaces registrados.</p> <p>El usuario ha de seleccionar uno de los items y pulsar "Ok".</p> <p>Seguidamente, el sistema se actualizará con la información del nuevo workspace seleccionado, cargando los benchmarks y resultados registrados en él.</p> <p>También podrá cancelar la acción, pulsando "Cancel".</p>
ID	UC12
Stereotypes	UseCase

Cambiar de workspace

1. Acceder a la opción de menú "Preferences -> Workspaces"
2. Seleccionar del desplegable "Workspaces" un workspace distinto al actual.
3. Seleccionar "Guardar".
4. **SYSTEM** Se crea la propiedad **worspace.change** en `\.isbench\isbench.properties`
5. **SYSTEM** Se muestra un mensaje informando de que el cambio se hará la próxima vez que se arranque la aplicación.

Cancelar el cambio de workspace

1. Acceder a la opción de menú "Preferences -> Workspaces"
2. **SYSTEM** Se muestra la ventana "Workspaces"
3. Seleccionar del desplegable "Workspaces" un workspace distinto al actual.
4. Seleccionar "Cancelar".

Aplicar cambio de workspace

1. Arrancar la aplicación.
2. **SYSTEM** El sistema comprueba que existe la propiedad **worspace.change** en `\.isbench\isbench.properties`.
3. **SYSTEM** Se comprueba que existe el archivo `workspace.xml` en el directorio que indica la

propiedad **worspace.change** .

4. **SYSTEM** Actualiza la propiedad **worspace.default** en el mismo archivo con el valor de **worspace.change** y borra esta última

5. **SYSTEM** Se carga la aplicación con el workspace actualizado.

Aplicar cambio de workspace incorrecto

1. Arrancar la aplicación.

2. **SYSTEM** El sistema comprueba que existe la propiedad **worspace.change** en \.isbench\isbench.properties.

3. **SYSTEM** Se comprueba que NO existe el archivo workspace.xml en el directorio que indica la propiedad **worspace.change** .

4. **SYSTEM** Muestra un mensaje de error indicando que el workspace actual no es correcto y muestra la ventana Workspaces para que se registre correctamente.

● Crear workspace

Name	Value
Description	<p>1 Workspace</p> <p>Un <i>workspace</i> será una ubicación física donde se guardarán archivos relacionados para su uso en IS Bench.</p> <p>Podrá contener:</p> <ul style="list-style-type: none">• Registro de algoritmos y benchmarks.• Archivos que servirán de entrada a los algoritmos a ejecutar.• Salidas de los algoritmos ejecutados.• Otras preferencias del usuario, como el idioma. <p>Se definirá un <i>workspace</i> por defecto, que podrá ser modificado por el usuario.</p> <p>2 Workspace por defecto</p> <p>Inicialmente, la aplicación tomará como <i>workspace</i> por defecto la carpeta /.isbench/default.</p> <p>Esta carpeta es la localización inicial que abrirán los selectores de entrada y salida y contendrá dos carpetas: <i>inputs</i> y <i>outputs</i>.</p>

Name	Value
	<p>La carpeta <i>inputs</i> será el directorio que se abrirá por defecto al seleccionar la entrada para la ejecución de un algoritmo.</p> <p>La carpeta <i>outputs</i> será el directorio que se abrirá por defecto al seleccionar la salida de la ejecución de un algoritmo.</p>
	<p>3 Crear un Workspace</p> <p>El usuario podrá crear un workspace accediendo a la ventana "Workspaces" desde la opción de menú "Preferences -> Workspaces".</p> <p>La ventana mostrará un deplegable combinado, en el que sus ítems serán los workspaces existentes en el archivo [home_usuario]/.isbench/isbench.properties.</p> <p>Podrá crear uno nuevo introduciendo el nombre de un directorio o seleccionándolo con el buscador (botón "...").</p> <p>El sistema creará una carpeta con el nombre introducido que contendrá un archivo workspace.xml. En este archivo se almacenará la configuración del workspace creado.</p> <p>Además el sistema preguntará al usuario si desea establecerlo como workspace actual. Si el usuario confirma, el cambio se hace efectivo la próxima vez que se ejecute la aplicación.</p>
ID	UC11
Stereotypes	UseCase
Justification	Creación de nuevos workspaces
Requirements	Crear un workspace, Implementación de ventana "Workspaces", Carga de workspaces registrados y sus preferencias

Crear un workspace

1. Seleccionar la opción de menú "Preferencias -> Workspaces".
2. **SYSTEM** Se muestra la ventana "Workspaces" con la información del workspace seleccionado actualmente.
3. Seleccionar un directorio con el botón *Examinar (...)*.
4. **SYSTEM** La etiqueta nombre se inicializa con el nombre del directorio y se convierte en editable.
5. Establecer las preferencias deseadas editando la tabla "Preferencias".
6. Seleccionar "Guardar".
7. **SYSTEM** Se crea el directorio seleccionado y en él las carpetas input, output y el archivo preferences.properties.

8. **SYSTEM** Se actualiza el archivo [user_home]\.isbench\isbench.properties añadiendo el nuevo workspace: workspace.nombre=ruta_absoluta_dir

9. **SYSTEM** El sistema pregunta si se desea establecer éste como el workspace actual.

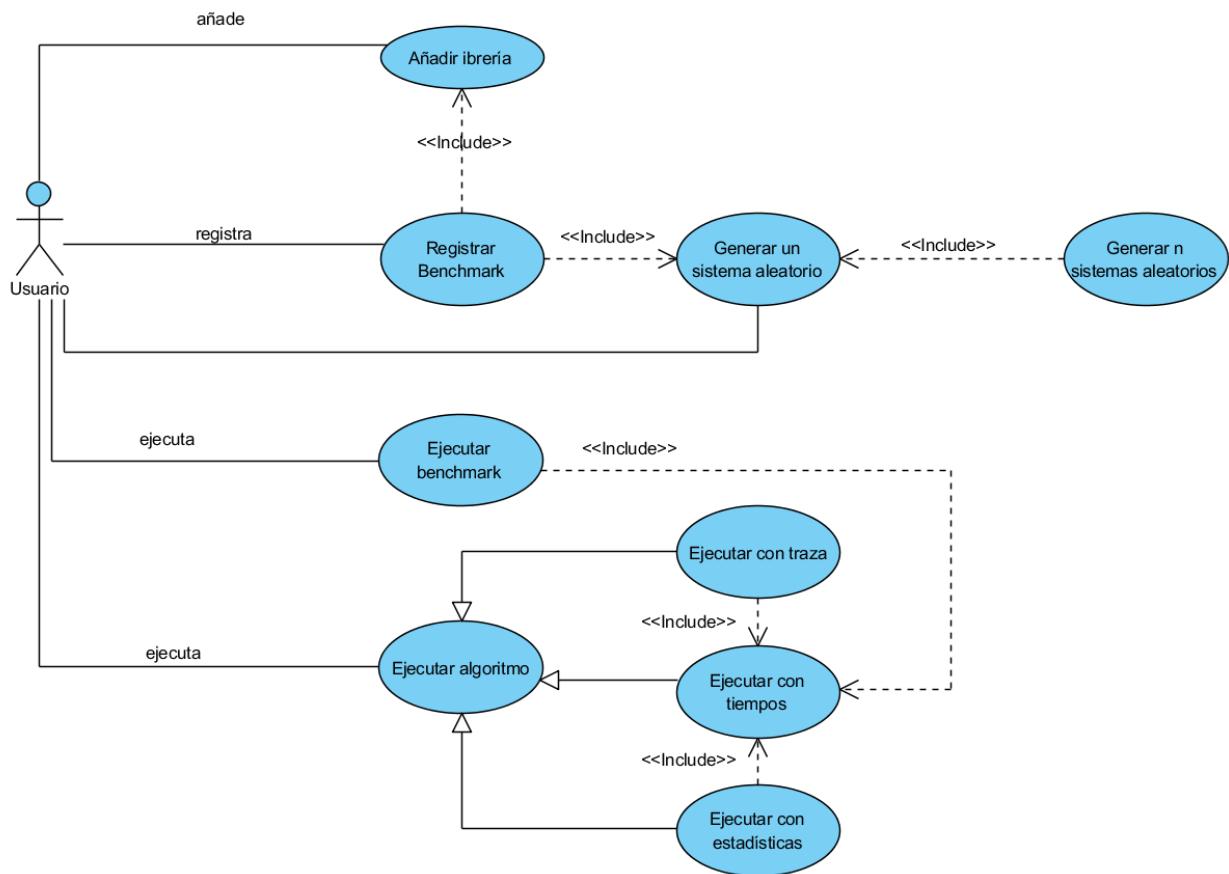
10. **if** se acepta establecerlo como workspace actual

10.1. **SYSTEM** El sistema actualiza el archivo \.isbench\isbench.properties estableciendo la propiedad workspace.change=workspace.nombre

10.2. **SYSTEM** Se muestra un mensaje informando de que el cambio se hará la próxima vez que se arranque la aplicación.

end if

6.2.3. Benchmarks



Summary

Name	Description
>User	Usuario general de la aplicación. No existirán roles ni permisos especiales. Cualquier usuario puede realizar cualquier acción.
Añadir librería	Añadir al workspace librerías externas con implementaciones de algoritmos sobre sistemas implicacionales.
Registrar Benchmark	Registro de benchmarks en el workspace.
Generar un sistema aleatorio	Generación de un conjunto de implicaciones aleatorio.
Generar n sistemas aleatorios	Generación de n conjuntos de implicaciones aleatorios.
Ejecutar benchmark	Ejecución de un benchmark.
Ejecutar algoritmo	Ejecución de un algoritmo.
Ejecutar con estadísticas	Ejecución de algoritmos con el modo “Statistics” activo.
Ejecutar con tiempos	Ejecución de algoritmos con el modo “Time” activo.

Name	Description
● Ejecutar con traza	Ejecución de algoritmos con el modo “Trace” activo.

Details

● Usuario

Name	Value
Description	Usuario general de la aplicación. No existirán roles ni permisos especiales. Cualquier usuario puede realizar cualquier acción.
ID	AC01
Visibility	Public

● Añadir librería

Name	Value
Description	Desde IS Bench se podrán añadir al workspace librerías externas con implementaciones de algoritmos sobre sistemas implicacionales.
ID	UC09
Stereotypes	UseCase
Justification	Registro de algoritmos en el workspace para su ejecución.
Preconditions	Las librerías a añadir contienen implementaciones de algoritmos sobre sistemas implicacionales que cumplen la API definida.

Añadir una librería

1. Hacer click en el botón “+” de la lista de algoritmos
2. Seleccionar uno o varios ficheros JAR y hacer click en Aceptar.
3. SYSTEM: Copia los ficheros seleccionados al workspace actual.
4. SYSTEM: Recarga la lista de algoritmos con los encontrados en las librerías añadidas.

Añadir una librería ya existente

1. Hacer click en el botón “+” de la lista de algoritmos
2. Seleccionar uno o varios ficheros JAR y hacer click en Aceptar.
3. SYSTEM: Pregunta si se desea sobrescribir el fichero.
4. Confirmar la acción.
5. SYSTEM: Copia los ficheros seleccionados al workspace actual.
6. SYSTEM: Recarga la lista de algoritmos con los encontrados en las librerías añadidas.

Cancelar la adición de una librería ya existente

1. Hacer click en el botón “+” de la lista de algoritmos
2. Seleccionar uno o varios ficheros JAR y hacer click en Aceptar.
3. SYSTEM: Pregunta si se desea sobrescribir el fichero.
4. Cancelar la acción.

● Registrar Benchmark

Name	Value
Description	<p>En el sistema existirá una pantalla <i>Benchmarks</i> que constará de dos pestañas:</p> <ul style="list-style-type: none"> • Add • Run <p>Para el registro de benchmarks, el usuario deberá acceder a la pestaña "Add".</p> <p>En esta pestaña se mostrará un formulario con los siguientes campos:</p> <ul style="list-style-type: none"> • "Name": Nombre del Benchmark. • "Input": Directorio que contiene los sistemas implicacionales de entrada para todos los algoritmos del Benchmark. <p>El directorio de entrada por defecto es [workspace_actual]/[nombre_benchmark]/input.</p> <p>A continuación de este campo se muestra un botón con dos opciones: <i>Seleccionar directorio</i>, <i>Generar</i>.</p> <p><i>Seleccionar directorio</i>: abre un cuadro de diálogo para seleccionar un directorio con ficheros de entrada.</p> <p><i>Generar</i>: Abre el generador de implicaciones, para crear los ficheros de entrada.</p> <ul style="list-style-type: none"> • Sección "Algorithms": Consta de dos listas de algoritmos. <ul style="list-style-type: none"> ○ A la izquierda se mostrará la lista de algoritmos registrados en el workspace actual. Éstos podrán ser filtrados mediante un filtro en vivo situado en la parte superior de la lista. ○ A la derecha la lista de algoritmos que se incluirán en el benchmark que se está creando. Éstos se seleccionarán con doble click de la lista anterior. <p>Todos los campos serán obligatorios.</p>
ID	UC10
Stereotypes	UseCase
Requirements	Diseño de formulario de registro de benchmarks., Carga de lista de algoritmos registrados en el workspace actual., Selección de sistema aleatorio como entrada., Guardar benchmark.
Preconditions	● Registrar Algoritmo

Registrar un benchmark

1. Acceder a la pantalla "Nuevo Benchmark".
2. Introducir un nombre en el campo "Nombre".
3. Introducir la ruta absoluta de un archivo que contiene un sistema implicacional en el campo "Entrada".
4. Seleccionar al menos un algoritmo de la lista de algoritmos.
5. Seleccionar la opción "Guardar".

Registrar un benchmark con un sistema de entrada aleatorio

1. Acceder a la pantalla "Nuevo Benchmark".
2. Introducir un nombre en el campo "Nombre".
3. Seleccionar la opción "Random" del botón del campo Input.
4. El sistema abre una ventana con el Generador de implicaciones.
5. Introducir número de atributos e implicaciones.
6. Seleccionar "Generate".
7. Seleccionar "Save".
8. Cerrar la ventana del generador.
9. Seleccionar al menos un algoritmo de la lista de algoritmos.
10. Seleccionar la opción "Guardar".

Registrar un benchmark vacío

1. Acceder a la pantalla "Nuevo Benchmark".
2. Seleccionar la opción "Guardar".
3. El sistema muestra un mensaje de error indicando que hay campos vacíos.
4. Introducir un nombre en el campo "Nombre".
5. Introducir la ruta absoluta de un archivo que contiene un sistema implicacional en el campo "Entrada".
6. Seleccionar al menos un algoritmo de la lista de algoritmos.
7. Seleccionar la opción "Guardar".

Registrar un benchmark sin algoritmos

1. Acceder a la pantalla "Nuevo Benchmark".
2. Introducir un nombre en el campo "Nombre".
3. Introducir la ruta absoluta de un archivo que contiene un sistema implicacional en el campo "Entrada".
4. Seleccionar la opción "Guardar".
5. El sistema muestra un mensaje de error indicando que se debe seleccionar al menos un algoritmo.
6. Seleccionar al menos un algoritmo de la lista de algoritmos.

7. Seleccionar la opción "Guardar".

Registra un benchmark si sistema de entrada

1. Acceder a la pantalla "Nuevo Benchmark".
2. Introducir un nombre en el campo "Nombre".
3. Seleccionar al menos un algoritmo de la lista de algoritmos.
4. Seleccionar la opción "Guardar".
5. El sistema muestra un mensaje de error indicando que el campo "Entrada" no puede ser vacío.
6. Introducir la ruta absoluta de un archivo que contiene un sistema implicacional en el campo "Entrada".
7. Seleccionar la opción "Guardar".

Registra un benchmark sin nombre

1. Acceder a la pantalla "Nuevo Benchmark".
2. Introducir la ruta absoluta de un archivo que contiene un sistema implicacional en el campo "Entrada".
3. Seleccionar al menos un algoritmo de la lista de algoritmos.
4. Seleccionar la opción "Guardar".
5. El sistema muestra un mensaje de error indicando que el campo "Nombre" no puede ser vacío.
6. Introducir un nombre en el campo "Nombre".
7. Seleccionar la opción "Guardar".

Registrar un benchmark con un nombre existente

1. Acceder a la pantalla "Nuevo Benchmark".
2. Introducir el nombre de un benchmark existente en el workspace actual en el campo "Nombre".
3. Introducir la ruta absoluta de un archivo que contiene un sistema implicacional en el campo "Entrada".
4. Seleccionar al menos un algoritmo de la lista de algoritmos.
5. Seleccionar la opción "Guardar".
6. El sistema muestra un mensaje de confirmación del benchmark existente con el mismo nombre.
7. Seleccionar la opción "Ok".

Cancelar el registro de un benchmark con un nombre existente

1. Acceder a la pantalla "Nuevo Benchmark".
2. Introducir el nombre de un benchmark existente en el workspace actual en el campo

"Nombre".

3. Introducir la ruta absoluta de un archivo que contiene un sistema implicacional en el campo "Entrada".
4. Seleccionar al menos un algoritmo de la lista de algoritmos.
5. Seleccionar la opción "Guardar".
6. El sistema muestra un mensaje de confirmación del benchmark existente con el mismo nombre.
7. Seleccionar la opción "Cancelar".

● Generar un sistema aleatorio

Name	Value
Description	<p>El sistema permite generar sistemas implicacionales aleatorios, que sirvan de entrada a un algoritmo.</p> <p>Estos sistemas son generados a partir de una serie de parámetros que el usuario debe proporcionar:</p> <ul style="list-style-type: none">• Número de atributos. Obligatorio. Debe ser mayor que 0.• Número de implicaciones. Obligatorio. Debe ser mayor que 0.• Tipo de atributos: Se puede seleccionar entre numéricos (1, 2, 3, ...), alfábéticos (a, b, c, ...) y alfanuméricos (a1, a2, a3, ...). Por defecto será numérico.• Mínimo y Máximo número de atributos en la premisa. No es obligatorio y por defecto sólo se establece el mínimo a 1 atributo.• Mínimo y Máximo número de atributos en la conclusión. No es obligatorio y por defecto sólo se establece el mínimo a 1 atributo.• Salida: Obligatorio para guardar el sistema en fichero, pero no antes. Será la ruta absoluta del fichero en el que se guardará el sistema generado. <p>Una vez introducidos los parámetros, el usuario podrá generar el sistema y la aplicación previsualizará éste.</p> <p>El usuario podrá guardarlo en la ruta introducida.</p>
ID	UC01

Generar un sistema aleatorio con valores por defecto

1. Introducir el número de atributos.
2. Introducir el número de implicaciones.
3. Pulsar el botón "Generate".

4. **SYSTEM** Se genera un sistema con el número de atributos e implicaciones especificados.
5. **SYSTEM** El tipo de los atributos es numérico, el tamaño mínimo de las premisas y de las conclusiones es 1.

Generación de un sistema aleatorio con 0 atributos.

1. Introducir el número de implicaciones.
2. Pulsar el botón "Generate".
3. **SYSTEM** Se muestra un error indicando que el número de atributos ha de ser mayor que 0.

Generación de un sistema aleatorio con 0 implicaciones.

1. Intorducir el número de atributos.
2. Pulsar el botón "Generate".
3. **SYSTEM** Se muestra un error indicando que el número de implicaciones ha de ser mayor que 0.

Generación de un sistema aleatorio con el tipo de atributos a, b, c,...

1. Introducir el número de atributos.
2. Introducir el número de implicaciones.
3. Seleccionar el tipo a, b, c...
4. Pulsar el botón "Generate".
5. **SYSTEM** Se genera un sistema con el número de atributos e implicaciones especificados y el tipo de los atributos es *a, b, c...*

Generación de un sistema aleatorio acotando el tamaño de la premisa, tanto mínimo como máximo.

1. Introducir el número de atributos.
2. Introducir el número de implicaciones.
3. Introducir un tamaño mínimo y máximo para la premisa.
4. **if** el tamaño mínimo de la premisa es **mayor** que el tamaño máximo
 - 4.1. **SYSTEM** Se muestra un mensaje de error indicando que el rango es incorrecto.
5. **else**
 - 5.1. Pulsar el botón "Generate".
 - 5.2. **SYSTEM** Se genera un sistema con el número de atributos e implicaciones especificados.
 - 5.3. **SYSTEM** El tipo de los atributos es numérico.
 - 5.4. **SYSTEM** La longitud de las premisas está entre el mínimo y máximo introducidos.

end if

Generación de un sistema aleatorio acotando el tamaño de la conclusión, tanto mínimo como máximo.

1. Introducir el número de atributos.
 2. Introducir el número de implicaciones.
 3. Introducir un tamaño mínimo y máximo para la **conclusión**.
 4. **if** el tamaño mínimo de la conclusión es **mayor** que el tamaño máximo
 - 4.1. **SYSTEM** Se muestra un mensaje de error indicando que el rango es incorrecto.
 5. **else**
 - 5.1. Pulsar el botón "Generate".
 - 5.2. **SYSTEM** Se genera un sistema con el número de atributos e implicaciones especificados.
 - 5.3. **SYSTEM** El tipo de los atributos es numérico.
 - 5.4. **SYSTEM** La longitud de las c está entre el mínimo y máximo introducidos.
- end if**

Guardar el sistema generado en un archivo.

1. Introducir el número de atributos.
2. Introducir el número de implicaciones.
3. Pulsar el botón "Generate".
4. **SYSTEM** Se genera un sistema con el número de atributos e implicaciones especificados.
5. Introducir la ruta absoluta del fichero en el que se desea guardar el sistema.
6. Pulsar el botón "Save".
7. **SYSTEM** Se crea / actualiza el archivo introducido con el sistema generado.

Generar n sistemas aleatorios

Name	Value
Description	Generación de n conjuntos de implicaciones aleatorios.
ID	UC19
Stereotypes	UseCase

Generar n sistemas aleatorios

1. Introducir el número de atributos.
2. Introducir el número de implicaciones.
3. Introducir un valor mayor que 1 en el número de sistemas a generar.
4. **SYSTEM** El botón "Generate" se deshabilita.

5. Introducir la ruta absoluta del directorio en el que se desea guardar los sistemas generados.
6. Introducir el nombre base de los sistemas.
7. **SYSTEM** El botón "Generate" se habilita.
8. Pulsar el botón "Generate".
9. **SYSTEM** Se genera el número de sistemas introducido, con el número de atributos e implicaciones especificados.
10. **SYSTEM** Se guardan los sigemas en el directorio introducido.
11. **SYSTEM** Los nombres de los archivos es el nombre introducido como nombre base, con el prefijo _1, _2, etc.
12. **SYSTEM** Se muestra un mensaje indicando el número de sistemas generados y dónde se han guardado.

Ejecutar benchmark

Name	Value
Description	<p>En el sistema existirá una pantalla <i>Benchmarks</i> que constará de dos zonas:</p> <ul style="list-style-type: none"> • Añadir • Ejecutar <p>Para la ejecución de benchmarks, el usuario deberá acceder a la zona "Ejecutar".</p> <p>Esta zona se dividirá a su vez en otras dos:</p> <ul style="list-style-type: none"> • A la izquierda se mostrarán los Benchmarks registrados en un árbol de dos niveles, en el que el primer nivel contendrá el nombre del Benchmark y en el segundo nivel, los algoritmos que lo componen. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> ▼ Basis Bench <div style="margin-top: 5px; font-size: 0.8em;"> Direct Optimal Basis Direct Optimal Basis II </div> </div> <ul style="list-style-type: none"> • A la derecha se mostrarán los campos de entrada y salida que contendrán las rutas del archivo con el sistema implicacional de entrada y el destino de los resultados de la ejecución correspondientemente. <p>Además, se mostrá un visor de resultados en el que se podrán cargar los archivos generados por la ejecución.</p> <p style="text-align: right;">El campo “Entrada” se inicializará con el archivo de</p>

Name	Value
	<p>entrada definido para el benchmark seleccionado y el campo "Salida" se inicializará con el directorio //output.</p> <p>Para ejecutar un benchmark el usuario seleccionará uno del árbol y pulsará el botón "Ejecutar".</p> <p>El modo de ejecución que se permite es "Time", generando sólo como salida los sistemas resultantes de cada algoritmo y los tiempos de ejecución de éstos.</p> <p>El sistema ejecutará cada uno de los algoritmos que componen el benchmark seleccionado, generando las salidas en el directorio que se indica en el campo "Salida":</p> <ul style="list-style-type: none"> • Archivos [abreviatura_alg]_output.txt con los sistemas implicacionales resultantes por cada algoritmo ejecutado, , siendo [abreviatura_alg] la abreviatura de cada algoritmo. • Archivos [abreviatura_alg]_history.txt con los tiempos de ejecución por cada algoritmo ejecutado, siendo [abreviatura_alg] la abreviatura de cada algoritmo. • Resumen de la ejecución, contiendo éste para cada algoritmo: <ul style="list-style-type: none"> ○ Algoritmo ejecutado. ○ Fecha y hora. ○ Fichero de salida ○ Tiempo de ejecución. <p>Los datos de este resumen son los que se mostrará cuando el usuario consulte los resultados de las ejecuciones.</p>
ID	UC08
Stereotypes	UseCase
Justification	Ejecución de conjunto de algoritmos para la posterior comparación de sus resultados.
Preconditions	 Registrar Benchmark  Registrar Algoritmo

Ejecución de un benchmark

1. Desde la pantalla principal (Home) hacer click en el botón Benchmarks.
2. Seleccionar un Benchmark en el árbol.
3. Hacer click en el botón "Run".
4. **SYSTEM** Se genera un archivo de salida por algoritmo y otro .log con los tiempos de ejecución.

Ejecución de un benchmark sin tiempos

1. Seleccionar un Benchmark en el árbol.
2. Desactivar la casilla "Time".
3. **SYSTEM** Hacer click en el botón "Run".

Ejecutar algoritmo

Name	Value
Description	<p>El sistema permitirá ejecutar un algoritmo previamente registrado.</p> <p>El usuario deberá proporcionar:</p> <ul style="list-style-type: none"> • Un sistema implicacional de entrada sobre el que se aplicará el algoritmo seleccionado. Estos sistemas se podrán seleccionar desde un fichero en disco, o generándolo con el Generador de Implicaciones. • Un fichero de salida en el que se guardará el sistema implicacional resultante devuelto por el algoritmo. Por defecto, el fichero de salida será [workspace]/output/[abreviatura_alg]_output.txt, siendo [workspace] el workspace actual y [abreviatura_alg] el nombre corto del algoritmo. • Uno o varios modos de ejecución que serán: <ul style="list-style-type: none"> - Tiempos - Trazas - Estadísticas <p>Una vez ejecutado el algoritmo seleccionado con los parámetros introducidos, se generará un archivo con el sistema implicacional de salida y otros con las trazas según los modos de ejecución seleccionados.</p> <p>Además, se persistirá esta ejecución guardando:</p> <ul style="list-style-type: none"> • Fecha / hora de ejecución. • Algoritmo ejecutado. • Tiempo de ejecución (si el modo "Time" fue activado) • Rutas absolutas de los ficheros que contienen: sistema de salida, histórico (si el modo "Trace" ha sido activado) y estadísticas (si el modo "Statistics" ha sido activado).
ID	UC04
Stereotypes	UseCase
Requirements	Diseño de pestaña "Run" en el área Benchmarks., Cargar árbol de benchmarks a partir de los registrados en el workspace

Name	Value
	actual., Inicializar valores por defecto al seleccionar un benchmark, Inicializar valores por defecto al seleccionar un algoritmo, Mostrar cuadro de diálogo para selección de entrada / salida, Ejecutar el algoritmo seleccionado, Ejecutar el benchmark seleccionado, Guardar resultados de la ejecución de un algoritmo.
Preconditions	 Registrar Algoritmo

Ejecutar un algoritmo

1. Acceder a la pantalla "Ejecución de Benchmarks"
2. Seleccionar un algoritmo del árbol de benchmarks.
3. **SYSTEM** Se incializa el campo "Output" con la ruta [workspace_actual]/output/[abreviatura_algoritmo].txt
4. Introducir el path de un archivo de salida.
5. Seleccionar los modos deseados.
6. Pulsar el botón "Ejecutar".
7. **SYSTEM** El sistema visualiza la traza según el modo de ejecución seleccionado.
8. **SYSTEM** Se guarda en la ruta de salida introducida, el sistema implicacional resultante.
9. **SYSTEM** En el mismo directorio que el archivo de salida, se guardan los ficheros de traza generados.
10. **SYSTEM** Se guarda el resultado de la ejecución para el algoritmo ejecutado en el benchmark al que pertenece el algoritmo.

Ejecutar un algoritmo con un fichero de entrada no existente

1. Seleccionar un algoritmo del árbol de benchmarks.
2. Introducir el path de un fichero no existente como entrada.
3. El sistema muestra un mensaje de error indicando que el fichero de entrada no existe.
4. Corregir el path del fichero de entrada con uno correcto.
5. Pulsar el botón "Ejecutar".
6. **SYSTEM** El sistema visualiza la traza según el modo de ejecución seleccionado.
7. **SYSTEM** Se guarda en la ruta de salida introducida, el sistema implicacional resultante.
8. **SYSTEM** En el mismo directorio que el archivo de salida, se guardan los ficheros de traza generados.
9. **SYSTEM** Se guarda el resultado de la ejecución para el algoritmo ejecutado en el benchmark al que pertenece el algoritmo.

Ejecutar con estadísticas

Name	Value
Description	<p>El usuario seleccionará el modo "Statistics" para la ejecución con traza.</p> <p>En esta ejecución, además de generar un archivo con el sistema implicacional de salida, el sistema genera el archivo [nombre_archivo_salida].csv, en el que se guardan la evolución de los tamaños del sistema implicacional procesado.</p> <p>[nombre_archivo_salida] es el nombre base del archivo seleccionado para la salida del algoritmo. P.e., si el archivo que se ha tomado como salida es <i>do_output.txt</i>, el archivo con los tiempos será <i>do.csv</i>.</p> <p>La información se guarda en archivos .csv para facilitar su visualización mediante tablas y gráficos.</p>
ID	UC17
Stereotypes	UseCase
Preconditions	 Registrar Algoritmo

Ejecutar algoritmo con estadísticas

1. 1. Acceder a la pantalla "Ejecución de Benchmarks"
 2. Seleccionar un algoritmo del árbol de benchmarks.
 3. Introducir el path de un fichero que contenga un sistema impilacional de entrada.
 4. Introducir el path de un archivo de salida.
 5. Seleccionar el modo "Statistics"
 6. Pulsar el botón "Ejecutar".
2. **SYSTEM** Se generar un archivo .log con el tiempo de ejecución y un archivo .csv con los tamaños.

● Ejecutar con tiempos

Name	Value
Description	<p>El usuario seleccionará el modo "Time" para la ejecución con tiempos.</p> <p>En esta ejecución, además de generar un archivo con el sistema implicacional de salida, el sistema genera el archivo [nombre_archivo_salida]_history.log, en el que se traza el tiempo de ejecución del algoritmo.</p> <p>[nombre_archivo_salida] es el nombre base del archivo seleccionado para la salida del algoritmo. P.e., si el archivo que se ha tomado como salida es <i>do_output.txt</i>, el archivo con los tiempos será <i>do_history.log</i>.</p>
ID	UC16
Stereotypes	UseCase
Requirements	Guardar el tiempo de ejecución del algoritmo en el resultado de éste. Mostrar el tiempo de ejecución en el visor.
Preconditions	 Registrar Algoritmo

Ejecutar un algoritmo con Tiempos

1. Acceder a la pantalla "Ejecución de Benchmarks"
2. Seleccionar un algoritmo del árbol de benchmarks.
3. **SYSTEM** Se incializa el campo "Output" con la ruta [workspace_actual]/output/[abreviatura_algoritmo].txt
4. Introducir el path de un archivo con un sistema implicacional de entrada.
5. Seleccionar el modo "Time".
6. Pulsar el botón "Ejecutar".
7. **SYSTEM** Se guarda en la ruta de salida introducida, el sistema implicacional resultante.
8. **SYSTEM** Se guarda el resultado de la ejecución para el algoritmo ejecutado en el benchmark al que pertenece el algoritmo y el tiempo de ejecución.
9. **SYSTEM** Se muestra en el visor el tiempo de ejecución.

Ejecutar con traza

Name	Value
Description	<p>El usuario seleccionará el modo "Trace" para la ejecución con traza.</p> <p>En esta ejecución, además de generar un archivo con el sistema implicacional de salida, el sistema genera el archivo [nombre_archivo_salida]_history.log, en el que se traza el tiempo de ejecución del algoritmo y traza de dicha ejecución.</p> <p>[nombre_archivo_salida] es el nombre base del archivo seleccionado para la salida del algoritmo. P.e., si el archivo que se ha tomado como salida es <i>do_output.txt</i>, el archivo con la traza será <i>do_history.log</i>.</p>
ID	UC15
Stereotypes	UseCase
Requirements	Escribir en un fichero la traza generada por un algoritmo., Mostrar la traza generada en el visor.
Preconditions	 Registrar Algoritmo

Ejecutar un algoritmo con traza y tiempos

1. Acceder a la pantalla "Ejecución de Benchmarks"
2. Seleccionar un algoritmo del árbol de benchmarks.
3. **SYSTEM** Se incializa el campo "Output" con la ruta [workspace_actual]/output/[abreviatura_algoritmo].txt
4. Introducir el path de un archivo con un sistema implicacional de entrada.
5. Seleccionar los modos "Time" y "Trace"
6. Pulsar el botón "Ejecutar".
7. **SYSTEM** El sistema visualiza la traza según el modo de ejecución seleccionado.
8. **SYSTEM** Se guarda en la ruta de salida introducida, el sistema implicacional resultante.
9. **SYSTEM** Se genera el archivo [nombre_archivo_salida]_history.log en el mismo directorio que el archivo de salida con la traza y tiempo de ejecución..
10. **SYSTEM** Se guarda el resultado de la ejecución para el algoritmo ejecutado en el benchmark al que pertenece el algoritmo.

6.2.4. Resultados



Summary

Name	Description
Usuario	<p>Usuario general de la aplicación.</p> <p>No existirán roles ni permisos especiales. Cualquier usuario puede realizar cualquier acción.</p>
Consultar resultados	<p>Para cada ejecución de un algoritmo se guardará un resumen de ésta que contendrá:</p> <ul style="list-style-type: none"> • Algoritmo ejecutado. • Fecha y hora. • Sistema de entrada. • Sistema de salida. • Fichero • Tiempo de ejecución. <p>El usuario podrá consultar estos datos a posteriori para comparar las distintas ejecuciones, en una tabla agrupada por benchmarks con n filas, una por algoritmo ejecutado y tres columnas:</p> <ul style="list-style-type: none"> - Tiempo de ejecución - Tamaño inicial - Tamaño final

Details

Usuario

Name	Value
Description	Usuario general de la aplicación. No existirán roles ni permisos especiales. Cualquier usuario puede realizar cualquier acción.
ID	AC01
Visibility	public

Consultar resultados

Name	Value
Description	Para cada ejecución de un algoritmo se guardará un resumen de ésta que contendrá: Algoritmo ejecutado. <ul style="list-style-type: none">• Fecha y hora.• Sistema de entrada.• Sistema de salida.• Fichero• Tiempo de ejecución. El usuario podrá consultar estos datos a posteriori para comparar las distintas ejecuciones, en una tabla agrupada por benchmarks con n filas, una por algoritmo ejecutado y tres columnas: <ul style="list-style-type: none">- Tiempo de ejecución- Tamaño inicial- Tamaño final
ID	UC02
Stereotypes	UseCase
Preconditions	 Ejecutar benchmark

Consulta de resultados de una ejecución

1. Seleccionar le botón "Results"
2. **SYSTEM** Se muestra una tabla en la que para cada benchmark ejecutado se muestra una fila por algoritmo y para cada uno su tiempo de ejecución, tamaño inicial del sistema y tamaño final.

6.3. Diagrama de componentes

En los diagramas de componentes se muestran los elementos de diseño de un sistema de software. Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces.

En el siguiente diagrama se muestran los principales componentes de la aplicación y las relaciones entre ellos.

- **is-bench:** Herramienta para ejecución de algoritmos y benchmarks sobre sistemas implicacionales.
- **is-algorithms:** API para la implementación y ejecución de algoritmos sobre sistemas implicacionales.
- **implications-generator:** Generador de sistemas implicacionales aleatorios.
- **java-lattice:** Librería de la Dra. K. Bertet, para el manejo de sistemas implicacionales.
- **trace.txt:** Fichero de traza que se genera en la ejecución de un algoritmo.
- **statistics.csv:** Fichero con las estadísticas de los resultados obtenidos en la ejecución de algoritmos.

6.4. Diagramas de paquetes

- **is-bench:** Librerías que implementan la herramienta para ejecución de algoritmos y benchmarks sobre sistemas implicacionales.
 - **is-bench-ui:** Paquete con las clases de la interfaz de usuario.
 - **is-bench-domain:** Paquete con las clases de dominio.
 - **is-bench-business:** Paquete con las clases relacionadas con la persistencia de la información en ficheros.
- **is-algorithms:** API para la implementación y ejecución de algoritmos sobre sistemas implicacionales.
- **implications-generator:** Contiene las clases que implementan el generador de sistemas implicacionales aleatorios.
- **is-commons:** Librería de utilidades genéricas.

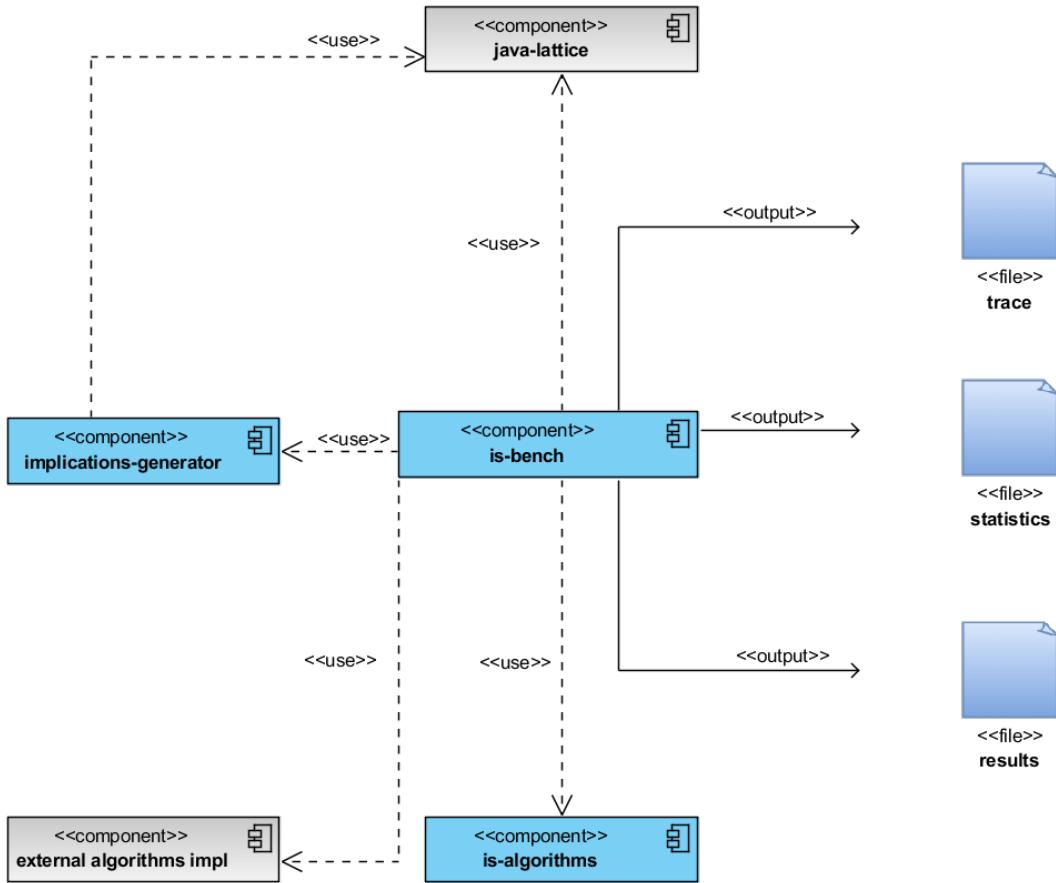


Figura 6.1: Diagrama de componentes

- **java-lattice**: Librería de la Dra. K. Bertet, para el manejo de sistemas implementacionales en la que se basan las librerías anteriores.
- **ch.qos.logback**: Librería de logging, para la generación de traza de la ejecución de algoritmos y benchmarks (Logback project).
- **guava**: Librería de Google que implementa diversas utilidades (Guava project). En este proyecto, se utiliza en concreto la implementación del bus de eventos (Eventbus).

6.5. Diagramas de clases

Los diagramas de clases muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras. Se dice que los diagramas de clases son diagramas «estáticos» porque muestran las clases, junto con sus métodos y atributos, así como las relaciones estáticas entre ellas: qué clases «conocen» a qué otras clases o qué clases «son parte» de otras clases, pero no muestran los métodos mediante los que se invocan entre ellas.

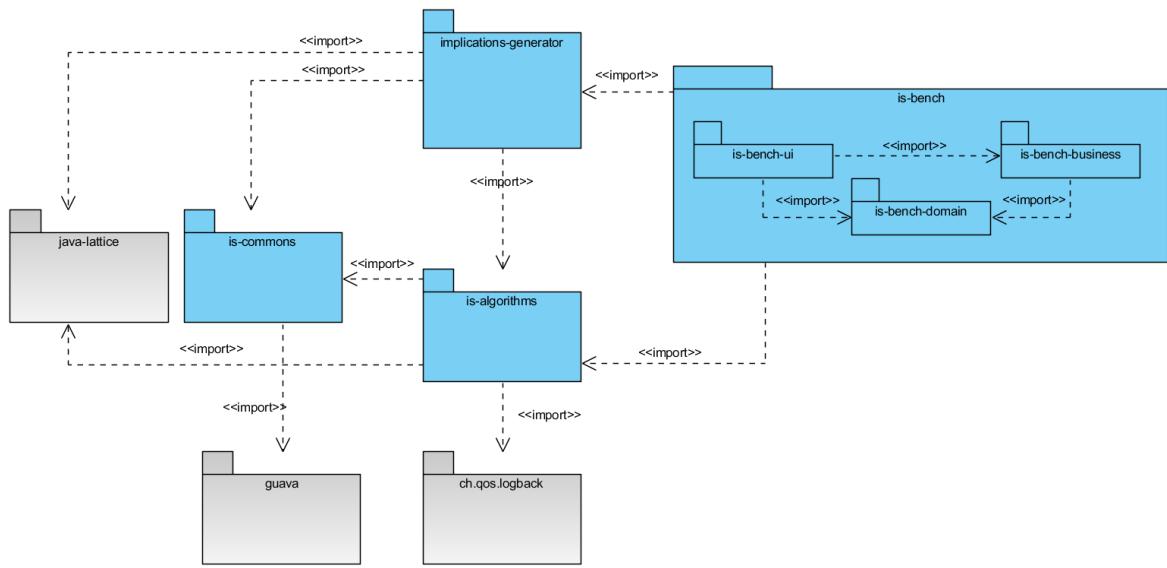


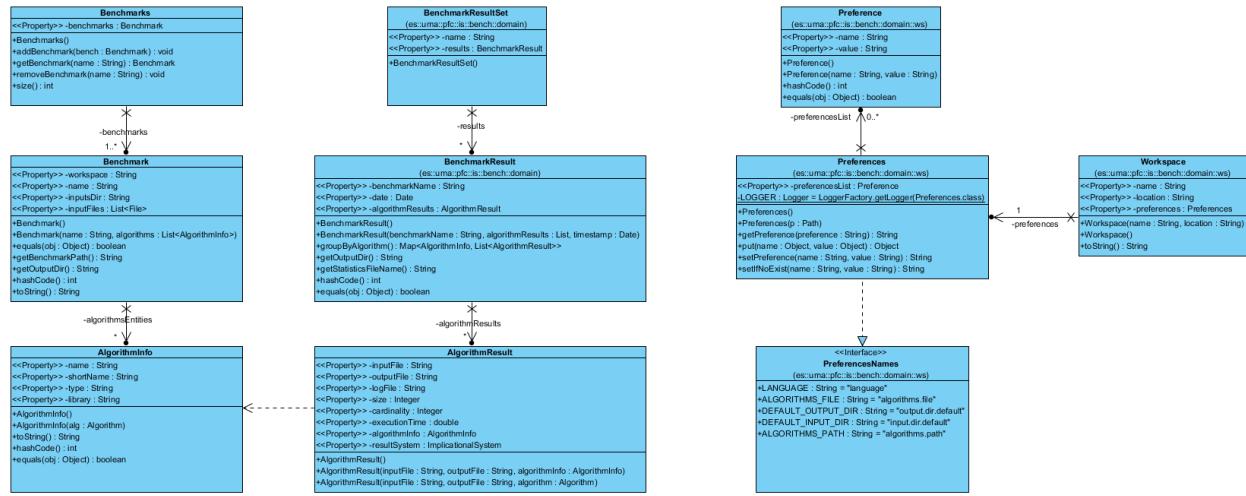
Figura 6.2: Diagrama de paquetes

En esta sección, se presentan los diagramas de clases asociados a cada uno de los casos de uso del apartado anterior y sus especificaciones.

6.5.1. Estereotipos

Name	Description
viewpoint	User interface.
controller	Controller of a FXML View.
FXML	FXML component injection.
listener	Method or class which handles component, model or eventbus events.
singleton	Implements Singleton pattern design.

6.5.2. Dominio



Summary

Name	Description
 AlgorithmInfo	Algorithm attributes.
 Benchmark	Benchmark.
 Benchmarks	Workspace benchmarks.
 AlgorithmResult	Algorithm result info.
 BenchmarkResult	Benchmark execution results.
 BenchmarkResultSet	Collection of benchmark results.
 Preference	User preference.
 Preferences	Contains the user preferences. These are serialized to properties file in the workspace.
 Workspace	Workspace.
 PreferencesNames	Preferences names.

Details

 AlgorithmInfo

Name	Value
Description	Entity with an algorithm attributes.
Visibility	public

Attributes

private name : String			
Description	Name.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		
private shortName : String			
Description	Short name.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		
private type : String			
Description	Algorithm implementation class name.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		
private library : String			
Description	Library.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		

Operations

public AlgorithmInfo ()	
Description	Constructor.

public AlgorithmInfo (alg : Algorithm)	
Parameters	alg
Description	Algorithm.
Multiplicity	1
Type	 Algorithm
Direction	inout
Java Detail	N/A
Description	Constructs an AlgorithmInfo instance from an algorithm implementation.

Benchmark

Name	Value
Description	Benchmark.
Visibility	public

Attributes

private workspace : String			
Description	Workspace.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		

private name : String			
Description	Name.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	false
Multiplicity	1		

private inputsDir : String			
Description	Input implicational systems directory path.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		

private inputFiles : java.util.List			
Description	Input files which will be copied to input dir.		
Stereotypes	Property		
Type	java.util.List		
Getter	true	Setter	true
Multiplicity	1..*		

private algorithmsEntities : List<AlgorithmInfo>			
Stereotypes	Property		
Type	 AlgorithmInfo		
Getter	true	Setter	false
Multiplicity	1..*		

Operations

public Benchmark ()	
Description	Constructor.

public Benchmark (name : String, algorithms : java.util.List)											
Parameters	<table border="1"> <tr> <td>name</td><td></td></tr> <tr> <td>Description</td><td>Name.</td></tr> <tr> <td>Multiplicity</td><td>1</td></tr> <tr> <td>Type</td><td> String</td></tr> <tr> <td>Direction</td><td>inout</td></tr> </table>	name		Description	Name.	Multiplicity	1	Type	String	Direction	inout
name											
Description	Name.										
Multiplicity	1										
Type	String										
Direction	inout										
	<table border="1"> <tr> <td>algorithms</td><td></td></tr> <tr> <td>Description</td><td>Algorithms.</td></tr> <tr> <td>Multiplicity</td><td>1..*</td></tr> <tr> <td>Type</td><td>java.util.List</td></tr> <tr> <td>Direction</td><td>inout</td></tr> </table>	algorithms		Description	Algorithms.	Multiplicity	1..*	Type	java.util.List	Direction	inout
algorithms											
Description	Algorithms.										
Multiplicity	1..*										
Type	java.util.List										
Direction	inout										
Description	Constructor.										
Exceptions	IllegalArgumentException if name or algorithms list is empty.										

public getBenchmarkPath () : String	
Description	The benchmark path.
Return Type Description	Benchmark path.

public getOutputDir () : String	
Description	Path of output dir of benchmark.
Return Type Description	Path.

Benchmarks

Name	Value
Description	Workspace benchmarks.
Visibility	public

Attributes

private benchmarks : Benchmark			
Stereotypes	Property		
Type	Benchmark		
Getter	true	Setter	true
Multiplicity	1..*		

Operations

public Benchmarks ()

Description	Constructor.
-------------	--------------

public addBenchmark (bench : Benchmark) : void

Parameters	bench	
	Description	Benchmark to add.
	Multiplicity	1
	Type	 Benchmark
	Direction	inout
Description	Adds a benchmark.	

public getBenchmark (name : String) : Benchmark

Parameters	name	
	Description	Name of benchmark to search.
	Multiplicity	1
	Type	 String
	Direction	inout
Description	Returns the benchmark with the name passed as parameter.	
Return Type Description	Bechmark found. Null if benchnmark not found.	

public removeBenchmark (name : String) : void

Parameters	name	
	Description	Name of benchmark to search.
	Multiplicity	1
	Type	 String
	Direction	inout
Description	Returns the benchmark with the name passed as parameter.	

public size () : int

Description	Returns the benchmarks count.
Return Type Descripiton	Benchmarks count.



AlgorithmResult

Name	Value
Description	Algorithm result info.
Visibility	public

Attributes

private inputFile : String			
Description	Implicational system input file.		
Stereotypes	Property		
Type	String		
Getter	true	Setter	false
Multiplicity	1..*		

private outputFile : String			
Description	Implicational system output file.		
Stereotypes	Property		
Type	String		
Getter	true	Setter	false
Multiplicity	1		

private logFile : String			
Description	Log file.		
Stereotypes	Property		
Type	String		
Getter	true	Setter	true
Multiplicity	1		

private size : Integer			
Description	Implicational System size.		
Stereotypes	Property		
Type	Integer		
Getter	true	Setter	false
Multiplicity	1		

private cardinality : Integer			
Description	Implicational System cardinality.		
Stereotypes	Property		
Type	Integer		
Getter	true	Setter	false
Multiplicity	1		

private executionTime : double			
Description	Execution time.		
Stereotypes	Property		
Type	 double		
Getter	true	Setter	true
Multiplicity	1		

private algorithmInfo : AlgorithmInfo			
Description	Info of the executed algorithm.		
Stereotypes	Property		
Type	 AlgorithmInfo		
Getter	true	Setter	false
Multiplicity	1		

private resultSystem : ImplicationalSystem			
Stereotypes	Property		
Type	 ImplicationalSystem		
Getter	true	Setter	false
Multiplicity	1		

Operations

public AlgorithmResult ()			
Description	Constructor.		
Leaf	false		

```
public AlgorithmResult (inputFile : String, outputFile : String,
algorithmInfo : AlgorithmInfo)
```

Parameters	inputFile	
	Description	Input system file.
	Multiplicity	1
	Type	● String
	Direction	inout
	outputFile	
	Description	Output system file.
	Multiplicity	1
	Type	● String
	Direction	inout
	outputFile	
	Description	Output system file.
	Multiplicity	1
	Type	● String
	Direction	inout
Description	Constructor.	

```
public AlgorithmResult (inputFile : String, outputFile : String, algorithm : Algorithm)
```

Parameters	inputFile	
	Description	Input system file.
	Multiplicity	1
	Type	● String
	Direction	inout
	outputFile	
	Description	Output system file.
	Multiplicity	1
	Type	● String
	Direction	inout
algorithm		
	Description	Algorithm.

public AlgorithmResult (inputFile : String, outputFile : String, algorithm : Algorithm)		
	Multiplicity	1
	Type	 Algorithm
	Direction	inout
Description	Constructor.	

BenchmarkResult

Name	Value
Description	Benchmark execution results.
Visibility	public

Attributes

private benchmarkName : String			
Description	Benchmark name.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	true
Multiplicity	Unspecified		

private date : java.util.Date			
Description	Execution date/time.		
Stereotypes	Property		
Type	java.util.Date		
Getter	true	Setter	true
Multiplicity	1		

private algorithmResults : List<AlgorithmResult>			
Description	Algorithm results.		
Stereotypes	Property		
Type	 AlgorithmResult		
Getter	true	Setter	true
Multiplicity	0..*		

Operations

public BenchmarkResult ()	
Description	Constructor.
public BenchmarkResult (benchmarkName : String, algorithmResults : List, timestamp : Date)	
Parameters	benchmarkName
	Multiplicity
	Type
	Direction
	algorithmResults
	Multiplicity
	Type
	Direction
	timestamp
	Multiplicity
	Type
	Direction
Description	Constructor.

public groupByAlgorithm () : java.util.Map	
Description	Groups the results by algorithm.
Return Type Description	Results indexed by algorithms.

public getOutputDir () : String	
Description	Returns the path dir of results.
Return Type Description	Path output dir.

public getStatisticsFileName () : String

Description	Returns the statistics file name.
Return Type Description.	Statistics file name.

BenchmarkResultSet

Name	Value
Description	Collection of benchmark results.
Visibility	public

Attributes**private name : String**

Description	Benchmark name.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		

private results : BenchmarkResult

Description	Benchmark results.		
Stereotypes	Property		
Type	 BenchmarkResult		
Getter	true	Setter	true
Multiplicity	*		

Operations**public BenchmarkResultSet ()**

Description	Constructor.
-------------	--------------



Preference

Name	Value
Description	User preference.
Visibility	public

Attributes

private name : String			
Description	Name.		
Stereotypes	Property		
Type	String		
Getter	true	Setter	true
Multiplicity	1		

private value : String			
Description	Value.		
Stereotypes	Property		
Type	String		
Allow Empty Name	false		
Getter	true	Setter	true
Multiplicity	1		

Operations

public Preference ()	
Description	Constructor.

public Preference (name : String, value : String)	
Parameters	name
	Description Name.
	Multiplicity 1
	Type  String
	Direction inout
	value
	Description Value.
	Multiplicity 1
	Type  String
	Direction inout
Description	Constructor.
Upper	1
Ordered	false
Unique	true
Query	false



Preferences

Name	Value
Description	Contains the user preferences. These are serialized to properties file in the workspace.
Visibility	public

Attributes

private preferencesList : Preference			
Description	Preferences list.		
Stereotypes	Property		
Type	 Preference		
Getter	true	Setter	false
Multiplicity	0..*		

private LOGGER : org.slf4j.Logger			
Description	Logger.		
Initial Value	LoggerFactory.getLogger(Preferences.class)		
Type	org.slf4j.Logger		
Getter	false	Setter	false
Multiplicity	Unspecified		

Operations

public Preferences ()	
Description	Constructor.

public Preferences (p : java.nio.file.Path)	
Parameters	p
	Description Preferences file absol
	Multiplicity Unspecified
	Type java.nio.file.Path
	Direction inout
Description	Builds an instance based on an existent preferences file.

public getPreference (preference : String) : String	
Parameters	preference
	Description Preference name.
	Multiplicity Unspecified
	Type  String
	Direction inout
Description	Value of a preference.
Return Type Description	Value of a preference.

public put (name : Object, value : Object) : Object																					
Parameters	<table border="1"> <thead> <tr> <th colspan="2">name</th></tr> </thead> <tbody> <tr> <td>Description</td><td>Name.</td></tr> <tr> <td>Multiplicity</td><td>1</td></tr> <tr> <td>Type</td><td>Object</td></tr> <tr> <td>Direction</td><td>inout</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="2">value</th></tr> </thead> <tbody> <tr> <td>Description</td><td>Value.</td></tr> <tr> <td>Multiplicity</td><td>1</td></tr> <tr> <td>Type</td><td>Object</td></tr> <tr> <td>Direction</td><td>inout</td></tr> </tbody> </table>	name		Description	Name.	Multiplicity	1	Type	Object	Direction	inout	value		Description	Value.	Multiplicity	1	Type	Object	Direction	inout
name																					
Description	Name.																				
Multiplicity	1																				
Type	Object																				
Direction	inout																				
value																					
Description	Value.																				
Multiplicity	1																				
Type	Object																				
Direction	inout																				
Description	Creates or modify a preference.																				
Return Type Description	Previous value of modified preference.																				
Query	false																				

public setPreference (name : String, value : String) : String																									
Parameters	<table border="1"> <thead> <tr> <th colspan="2">name</th></tr> </thead> <tbody> <tr> <td>Description</td><td>Name of preference.</td></tr> <tr> <td>Multiplicity</td><td>Unspecified</td></tr> <tr> <td>Type</td><td>String</td></tr> <tr> <td>Direction</td><td>inout</td></tr> <tr> <td>Java Detail</td><td>N/A</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="2">value</th></tr> </thead> <tbody> <tr> <td>Description</td><td>Value.</td></tr> <tr> <td>Multiplicity</td><td>Unspecified</td></tr> <tr> <td>Type</td><td>String</td></tr> <tr> <td>Direction</td><td>inout</td></tr> <tr> <td>Java Detail</td><td>N/A</td></tr> </tbody> </table>	name		Description	Name of preference.	Multiplicity	Unspecified	Type	String	Direction	inout	Java Detail	N/A	value		Description	Value.	Multiplicity	Unspecified	Type	String	Direction	inout	Java Detail	N/A
name																									
Description	Name of preference.																								
Multiplicity	Unspecified																								
Type	String																								
Direction	inout																								
Java Detail	N/A																								
value																									
Description	Value.																								
Multiplicity	Unspecified																								
Type	String																								
Direction	inout																								
Java Detail	N/A																								
Description	Sets the value of a preference. If value is null, remove the preference if exists.																								

public setIfNotExist (name : String, value : String) : String		
Parameters	name	
	Description	Name of preference.
	Multiplicity	1
	Type	● String
	Direction	inout
	value	
	Description	Value.
	Multiplicity	1
	Type	● String
	Direction	inout
Description	Sets the value of a preference if no exists. If value is null, remove the preference if exists.	

Workspace

Name	Value
Description	Workspace.
Visibility	public

Attributes

private name : String			
Description	Name.		
Stereotypes	Property		
Type	● String		
Getter	true	Setter	true
Multiplicity	1		

private location : String			
Description	Location.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		

private preferences : Preferences			
Description	Preferences.		
Stereotypes	Property		
Type	 Preferences		
Getter	true	Setter	true
Multiplicity	1		

Operations

public Workspace (name : String, location : String)			
Parameters	name		
	Multiplicity	1	
	Type	 String	
	Direction	inout	
	location		
	Multiplicity	1	
	Type	 String	
	Direction	inout	
	Description	Constructor.	

public Workspace ()			
Description			
Constructor.			



PreferencesNames

Name	Value
Description	Preferences names.
Visibility	public
Stereotypes	Interface

Attributes

public LANGUAGE : String			
Description	Language preference name.		
Initial Value	"language"		
Type	String		
Getter	false	Setter	false
Derived	false		
Multiplicity	1		

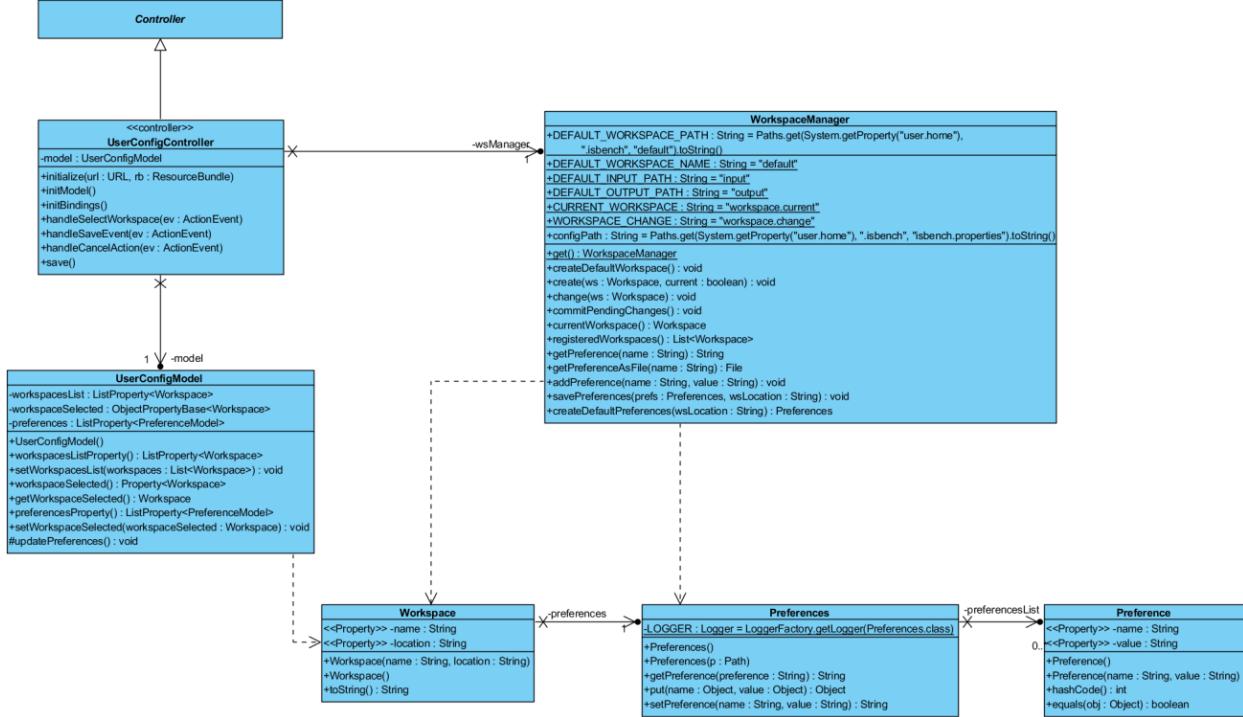
public ALGORITHMS_FILE : String			
Description	Language preference name.		
Initial Value	"algorithms.file"		
Type	String		
Getter	false	Setter	false
Multiplicity	1		

public DEFAULT_OUTPUT_DIR : String			
Description	Default output dir.		
Initial Value	"output.dir.default"		
Type	String		
Getter	false	Setter	false
Multiplicity	1		

public DEFAULT_INPUT_DIR : String			
Description	Default input dir.		
Initial Value	"input.dir.default"		
Type	String		
Getter	false	Setter	false
Multiplicity	1		

public ALGORITHMS_PATH : String			
Description	Algorihtms libraries path.		
Initial Value	"algorithms.path"		
Type	String		
Getter	false	Setter	false
Multiplicity	1		

6.5.3. Workspaces



Summary

Name	Description
Controller	Controller of a FXML View.
WorkspaceManager	Workspaces manager.
UserConfigController	Workspaces view controller.
UserConfigModel	Workspaces model.
Preference	User preference.
Workspace	Workspace.
Preferences	Contains the user preferences. These are serialized to properties file in the workspace.

Details

WorkspaceManager

Name	Value
Description	Workspaces manager.
Visibility	public

Attributes

<u>private LOGGER : org.slf4j.Logger</u>			
Description	Logger.		
Initial Value	LoggerFactory.getLogger(WorkspaceManager.class)		
Type	org.slf4j.Logger		
Getter	false	Setter	false
Multiplicity	1		

<u>public DEFAULT_WORKSPACE_PATH : String</u>			
Description	Default workspace location.		
Initial Value	Paths.get(System.getProperty("user.home"), ".isbench", "default").toString()		
Type	String		
Getter	false	Setter	false
Multiplicity	1		

<u>public DEFAULT_WORKSPACE_NAME : String</u>			
Description	Default workspace name.		
Initial Value	"default"		
Type	String		
Getter	false	Setter	false
Multiplicity	1		

<u>public DEFAULT_INPUT_PATH : String</u>			
Description	Input default directory.		
Initial Value	"input"		
Type	String		
Getter	false	Setter	false
Multiplicity	1		

<u>public DEFAULT_OUTPUT_PATH : String</u>			
Description	Output default directory.		
Initial Value	"output"		
Type	String		
Getter	false	Setter	false
Multiplicity	1		

<u>public CURRENT_WORKSPACE : String</u>			
Description	Current workspace property.		
Initial Value	"workspace.current"		
Type	String		
Getter	false	Setter	false
Multiplicity	1		

<u>public WORKSPACE_CHANGE : String</u>			
Description	Current workspace property.		
Initial Value	"workspace.change"		
Type	String		
Getter	false	Setter	false
Multiplicity	1		

<u>private config : java.util.Properties</u>			
Description	General configuration.		
Type	java.util.Properties		
Getter	false	Setter	false
Derived	false		
Multiplicity	1		

public configPath : String			
Description	Configuration file path.		
Initial Value	Paths.getProperty("user.home"), ".isbench", "isbench.properties").toString()		
Type	 String		
Getter	false	Setter	false
Multiplicity	1		

private me : WorkspaceManager			
Description	Single instance.		
Type	 WorkspaceManager		
Getter	false	Setter	false
Multiplicity	1		
Is ID	false		
Leaf	false		

private current : Workspace			
Description	Current workspace.		
Type	 Workspace		
Getter	false	Setter	false
Multiplicity	1		

Operations

protected WorkspaceManager ()	
Description	Constructor. Protected constructor. For gets an instance of WorkspaceManger, will must be use the get() method.

protected WorkspaceManager (configPath : String, config : java.util.Properties)	
Parameters	
	configPath
Description	Configuration file path.
Multiplicity	1
Type	String
Direction	inout
	config
Description	Configuration properties file.
Multiplicity	1
Type	java.util.Properties
Direction	inout
Description	Protected constructor. For gets an instance of WorkspaceManger, will must be use the get() method.

<u>public get () : WorkspaceManager</u>	
Description	
Description	Returns a single instance of WorkspaceManager.
Return Type Description	WorkspaceManager single instance.

<u>protected initialize () : void</u>	
Description	
Description	<p>Initializes the workspacemanager.</p> <p>If no exists the config file, creates it.</p> <p>If the current workspace not is setted, creates the default workspace and sets it as the current.</p>

<u>protected initConfig () : void</u>	
Description	
Description	Loads the isbench.properties file and if no exists, creates it.
Exceptions	IOException if read / Write error occur.

protected setCurrentWorkspace (ws : Workspace) : void	
Parameters	ws
	Multiplicity
	Type
	Direction
Description	Only for testing purpose.

public createDefaultWorkspace () : void	
Description	Creates the default workspace.

public create (ws : Workspace, current : boolean) : void	
Parameters	ws
	Description
	Multiplicity
	Type
Description	Creates the basic workspace directories and files hirerchachy, based on the info that contains the ws param.

public currentWorkspace () : Workspace	
Description	Returns the current workspace.
Return Type Description	Current workspace.

public registeredWorkspaces () : java.util.List<Workspace>	
Description	Returns the registered workspaces list.
Return Type Description	Workspaces list.

public change (ws : Workspace) : void	
Parameters	ws
	Description
	Multiplicity
	Type
	Direction
Description	Mark a workspace as current. The change will take effect with commitPendingChanges() method.

public commitPendingChanges () : void	
Description	The workspace change takes effect.

public getPreference (name : String) : String	
Parameters	name
	Description
	Multiplicity
	Type
	Direction
Description	Gets a preference value of the current workspace.
Return Type Description	Preference value. null if the preference not exists.

public getPreferenceAsFile (name : String) : java.io.File	
Parameters	name
	Description
	Multiplicity
	Type
	Direction
Description	Assume that a preference value is the file path and returns as it.
Return Type Description	File given from the consulted preference.
Exceptions	InvalidPathException if the path string cannot be converted to a Path.

public addPreference (name : String, value : String) : void	
Parameters	name
	Description Name.
	Multiplicity 1
	Type  String
	Direction inout
	value
	Description Value.
	Multiplicity 1
	Type  String
	Direction inout
Description	Adds a new preference.

public savePreferences (prefs : Preferences, wsLocation : String) : void	
Parameters	prefs
	Description Preferences.
	Multiplicity 1
	Type  Preferences
	Direction inout
	wsLocation
	Description Directory of the preferences.properties file.
	Multiplicity 1
	Type  String
	Direction inout
Description	Saves the preferences in a preferences.properties file, in a path (wsLocation param).
Exceptions	IOException if an error saving the file occur.

public createDefaultPreferences (wsLocation : String) : Preferences	
Parameters	wsLocation
	Multiplicity
	Type
	Direction
Description	Create a default preferences.
Return Type Description	Preferences created.

protected saveWorkspace (ws : Workspace, current : boolean) : void	
Parameters	ws
	Description
	Multiplicity
	Type
	Direction
	current
	Description
	Multiplicity
	Type
	Direction
Description	Save workspace in config file.
Exceptions	IOException if an error saving the file occur.

protected saveConfig () : void	
Description	Save the config in the file.
Exceptions	IOException if an error saving the file occur.

UserConfigController

Name	Value
Description	Workspaces view controller.
Visibility	public
Stereotypes	controller

Attributes

private model : UserConfigModel			
Description	Workspaces model.		
Type	 UserConfigModel		
Getter	false	Setter	false
Multiplicity	1		

private wsManager : WorkspaceManager			
Description	Workspaces manager.		
Type	 WorkspaceManager		
Getter	false	Setter	false
Multiplicity	1		

Operations

public initialize (url : java.net.URL, rb : java.util.ResourceBundle)			
Parameters	url		
	Description	View URL.	
	Multiplicity	1	
	Type	java.net.URL	
	Direction	inout	
	rb		
	Description	Internationalization resources.	
	Multiplicity	1	
	Type	java.util.ResourceBundle	
	Direction	inout	
Description	Initializes the controller class.		

public initModel ()			
Description			
Initializes the model with the registered workspaces.			

public initBindings ()			
Description			
Initializes the view-model bindings.			

public handleSelectWorkspace (ev : ActionEvent)		
Parameters	ev	
	Description	Action event.
	Multiplicity	1
	Type	ActionEvent
	Direction	inout
Description	Handles the event thrown when <i>Search(...)</i> button is pressed. Open a dialog for directory selection. The selection shows in the Location field.	
Stereotypes	FXML	

public handleSaveEvent (ev : ActionEvent)		
Parameters	ev	
	Description	Action event.
	Multiplicity	1
	Type	ActionEvent
	Direction	inout
Description	Handles the event thrown when Save button is pressed. Invokes the save() method to save the changes and closes	
Stereotypes	FXML	

public handleCancelAction (ev : ActionEvent)		
Parameters	ev	
	Description	Action event.
	Multiplicity	1
	Type	ActionEvent
	Direction	inout
Description	Ignores the changes and closes the window.	
Stereotypes	FXML	

public save()	
Description	Saves the workspace change. If the workspace is new, asks to the user if he would like establish it as the current workspace. If the user accepts, shows a new message indication that the change will take effect in the next application start.

UserConfigModel

Name	Value
Description	UserConfig view model.
Visibility	public

Attributes

private workspacesList : javafx.beans.property.ListProperty			
Description	Workspaces list.		
Type	javafx.beans.property.ListProperty		
Getter	false	Setter	false
Multiplicity	1		

private workspaceSelected : javafx.beans.property.ObjectPropertyBase			
Description	Workspace selected binding property.		
Type	javafx.beans.property.ObjectPropertyBase		
Getter	false	Setter	false
Multiplicity	1		

private preferences : javafx.beans.property.ListProperty			
Description	Preferences of selected workspace binding property.		
Type	javafx.beans.property.ListProperty		
Getter	false	Setter	false
Multiplicity	1		

Operations

public UserConfigModel ()	
Description	Constructor.

public setWorkspacesList (workspaces : java.util.List<Workspace>) : void		
Parameters	workspaces	
	Description	Workspaces paths.
	Multiplicity	0..*
	Type	java.util.List<Workspace>
	Direction	inout
Description	Sets the workspaces list.	

public getWorkspaceSelected () : Workspace	
Description	Gets the selected workspace from workspaceSelected property.
Return Type Description	The selected workspace.

protected updatePreferences () : void	
Description	Updates the preferences with the selected workspace's preferences.

public setWorkspaceSelected (workspaceSelected : Workspace) : void	
Parameters	workspaceSelected
	Description The workspaceSelected to set.
	Multiplicity 1
	Type  Workspace
	Direction inout
Description	Sets the selected workspace.

Preference

Name	Value
Description	User preference.
Visibility	public

Attributes

private name : String			
Description	Name.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		

private value : String			
Description	Value.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		

Operations

public Preference ()

Description	Constructor.
-------------	--------------

public Preference (name : String, value : String)

name	
Description	Name.
Multiplicity	1
Type	● String
Direction	inout
value	
Description	Value.
Multiplicity	1
Type	● String
Direction	inout
Description	Constructor.

public hashCode () : int

Description	Compute the hash code based on the name property.
Return Type Description	Hash code.

public equals (obj : Object) : boolean

Parameters		obj
Description	Object to compare.	
Multiplicity	1	
Type	Object	
Direction	inout	
Description	Two Preference instances are equal if their names are.	
Return Type Description	True if the obj parameter is a Preference instance, and its names is equal to this instance name.	



Workspace

Name	Value
Description	Workspace.
Visibility	public

Attributes

private name : String			
Description	Name.		
Stereotypes	Property		
Type	String		
Getter	true	Setter	true
Multiplicity	1		

private location : String			
Description	Location.		
Stereotypes	Property		
Type	String		
Getter	true	Setter	true
Multiplicity	1		

private preferences : Preferences			
Description	User preferences in the workspace.		
Stereotypes	Property		
Type	Preferences		
Getter	true	Setter	true
Multiplicity	1		

Operations

public Workspace (name : String, location : String)		
Parameters	name	
	Multiplicity	1
	Type	String
	Direction	inout
	location	
	Multiplicity	1
	Type	String
	Direction	inout
Description	Creates a workspace with the name and location specified.	

public Workspace ()	
Description	Constructor.

public toString () : String	
Description	Workspace string representation, this is the absolute path of its location.
Return Type Description	Workspace string representation.

Preferences

Name	Value
Description	Contains the user preferences. These are serialized to properties file in the workspace.
Visibility	public

Attributes

private preferencesList : Preference			
Description	Preferences list.		
Stereotypes	Property		
Type	 Preference		
Getter	true	Setter	false
Multiplicity	0..*		

private LOGGER : org.slf4j.Logger			
Description	Logger.		
Initial Value	LoggerFactory.getLogger(Preferences.class)		
Type	org.slf4j.Logger		
Getter	false	Setter	false
Derived	false		
Multiplicity	1		

Operations

public Preferences ()	
Description	Constructor.

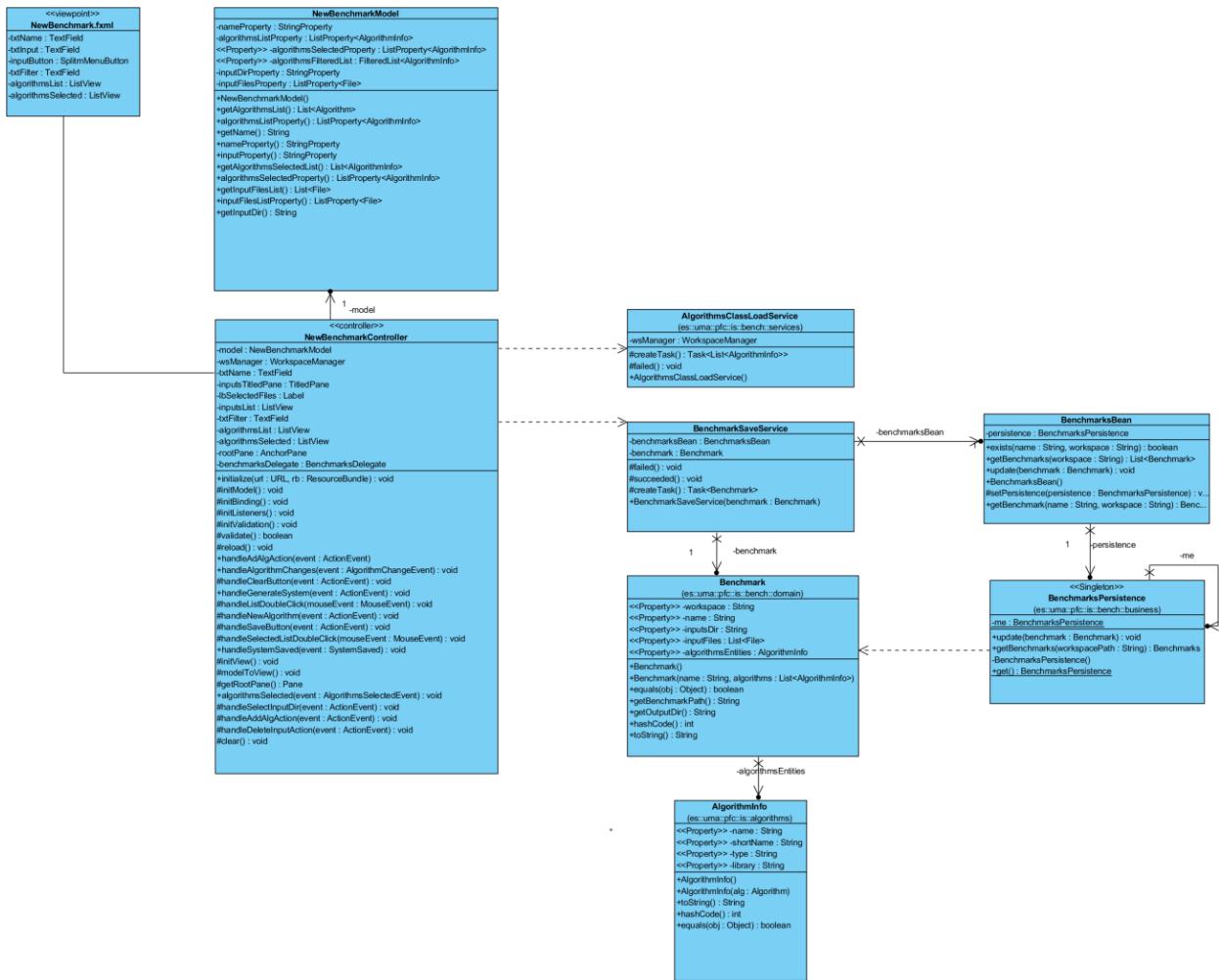
public Preferences (p : java.nio.file.Path)		
Parameters	p	
	Description	Preferences file absolute path.
	Multiplicity	1
	Type	java.nio.file.Path
	Direction	inout
Description	Builds an instance based on an existent preferences file.	

public getPreference (preference : String) : String		
Parameters	preference	
	Description	Preference name.
	Multiplicity	1
	Type	String
	Direction	inout
Description	Preference value.	
Return Type Description	Preference value.	

public put (name : Object, value : Object) : Object		
Parameters	name	
	Description	Name.
	Multiplicity	1
	Type	Object
	Direction	inout
	value	
	Description	Value.
	Multiplicity	1
	Type	Object
	Direction	inout
Description	Creates or modify a preference.	
Return Type Description	Previous value of modified preference.	

public setPreference (name : String, value : String) : String	
Parameters	
	name
	Multiplicity
	1
	Type
	String
	Direction
	inout
	value
	Description
	Value.
	Multiplicity
	1
	Type
	String
	Direction
	inout
Description	Sets the value of a preference. If value is null, remove the preference if exists.

6.5.4. Registrar Benchmarks



Summary

Name	Description
NewBenchmark.fxml	New benchmark view.
NewBenchmarkModel	New Benchmark view model.
AlgorithmsClassLoadService	Loads the algorithms found in the libraries included in the workspace lib folder.
NewBenchmarkController	NewBenchmark view Controller class.
BenchmarksBean	Business logic for insert, modify and delete algorithms.
BenchmarkSaveService	Service which saves benchmark changes.
Benchmark	Benchmark entity.
BenchmarksPersistence	Persist the benchmarks into an XML file entities using JAXB.
AlgorithmInfo	Entity with an algorithm attributes.

Description

Class diagram that shows the relation between the classes involved in the registry of benchmarks.

Details



NewBenchmark.fxml

Name	Value
Description	New benchmark view.
Visibility	public
Stereotypes	viewpoint

Attributes

private txtName : javafx.scene.control.TextField

Description	Name field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		

private txtInput : javafx.scene.control.TextField

Description	Absolute path of input implicational system file.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		

private inputButton : javafx.scene.control.SplitMenuItemButton

Description	Button for select the input implicational system file. It has two options: select the file or generate a random system.		
Type	javafx.scene.control.SplitMenuItemButton		
Getter	false	Setter	false
Multiplicity	1		

private txtFilter : javafx.scene.control.TextField			
Description	Search algorithms field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		

private algorithmsList : javafx.scene.control.ListView			
Description	Registered algorithms list in the current workspace.		
Type	javafx.scene.control.ListView		
Getter	false	Setter	false
Multiplicity	1		

private algorithmsSelected : javafx.scene.control.ListView			
Description	List which contains the selected algorithms from registered algorithms list.		
Type	javafx.scene.control.ListView		
Getter	false	Setter	false
Multiplicity	1		

NewBenchmarkController

Name	Value
Description	NewBenchmark view controller.
Visibility	public
Stereotypes	controller

Attributes

private model : NewBenchmarkModel			
Description	Model.		
Type	 NewBenchmarkModel		
Getter	false	Setter	false
Multiplicity	1		

private txtName : javafx.scene.control.TextField			
Description	Name field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private inputsTitledPane : javafx.scene.control.TitledPane			
Description	Container of inputs implicational systems selected.		
Type	javafx.scene.control.TitledPane		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private lbSelectedFiles : javafx.scene.control.Label			
Description	Label with selected files count.		
Type	javafx.scene.control.Label		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private inputsList : javafx.scene.control.ListView			
Description	List of selected input files.		
Type	javafx.scene.control.ListView		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private txtFilter : javafx.scene.control.TextField			
Description	Algorithms filter.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private algorithmsList : javafx.scene.control.ListView			
Description	Available algorithms.		
Type	ListView		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private algorithmsSelected : javafx.scene.control.ListView			
Description	Algorithms selected.		
Type	ListView		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private rootPane : javafx.scene.layout.AnchorPane			
Description	Root pane.		
Type	AnchorPane		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private wsManager: WorkspaceManager			
Description	Workspace manager.		
Type	 WorkspaceManager		
Getter	false	Setter	false
Multiplicity	1		

private benchmarksDelegate : BenchmarksDelegate

Description	Delegate for benchmarks business logic.		
Type	 BenchmarksDelegate		
Getter	false	Setter	false
Multiplicity	1		

Operations**public initialize (url : java.net.URL, rb : java.util.ResourceBundle) : void**

Parameters	url	
	Description	URL of the view.
	Multiplicity	1
	Type	java.net.URL
	Direction	inout
	rb	
	Description	Resource bundle.
	Multiplicity	1
	Type	java.util.ResourceBundle
	Direction	inout
Description	Initializes the controller class.	

protected initModel () : void

Description	Initializes the model.
-------------	------------------------

protected initBinding () : void

Description	Initializes the binding between view components and the model.
-------------	--

protected initListeners () : void

Description	Initializes the components view listeners.
-------------	--

protected initValidation () : void

Description	Initializes the validation support.
-------------	-------------------------------------

protected initView () : void	
Description	Initializes the inputsList selection mode.
Exceptions	IOException

protected modelToView () : void	
Description	Load the algorithms list with algorihtms of the model.

protected validate () : boolean	
Description	Form validations.
Return Type Description	true if there is one algorithm selected at least, and the benchmark name not exists or user wants override it, and parent validations is succeeded. False otherwise.

protected reload () : void	
Description	Reloads the view and model.

public handleAlgorithmChanges (event : AlgorithmChangeEvent) : void	
Parameters	event Description Event thrown when an algorithms is created or modified. Multiplicity 1 Type AlgorithmChangeEvent Direction inout
Description	Handles the AlgorithmChangeEvent published by the Eventbus. Reloads the model and view.

protected handleClearButton (event : javafx.event.ActionEvent) : void	
Parameters	event Description Action event. Multiplicity 1 Type javafx.event.ActionEvent Direction inout
Description	When the <i>Clear</i> button is pressed, the fields are cleared.
Stereotypes	FXML

public handleGenerateSystem (event : javafx.event.ActionEvent) : void

Parameters	event	
	Description	Event thrown when the <i>Random</i> option of <i>Input</i> button is pressed.
	Multiplicity	1
	Type	javafx.event.ActionEvent
	Direction	inout
Description	Shows the generator panel for generate a random system.	
Stereotypes	FXML	

protected handleListDoubleClick (mouseEvent : javafx.scene.input.MouseEvent) : void

Parameters	mouseEvent	
	Description	Mouse event.
	Multiplicity	1
	Type	javafx.scene.input.MouseEvent
	Direction	inout
Description	When there is a double click in algorithms list, the selection is added to algorithms selected.	
Stereotypes	FXML	

protected handleNewAlgorithm (event : javafx.event.ActionEvent) : void

Parameters	event	
	Description	Action event.
	Multiplicity	1
	Type	javafx.event.ActionEvent
	Direction	inout
Description	Handles the event thrown when <i>New Algorithm</i> button is pressed.	
Stereotypes	FXML	

protected handleSaveButton (event : javafx.event.ActionEvent) : void

Parameters	event	
	Description	Action event.
	Multiplicity	1
	Type	javafx.event.ActionEvent
Direction	inout	
Description	When the <i>Save</i> button is pressed, the current values are validated and saved.	
Stereotypes	FXML	

protected handleSelectedListDoubleClick (mouseEvent : javafx.scene.input.MouseEvent) : void

Parameters	mouseEvent	
	Description	Mouse event.
	Multiplicity	1
	Type	javafx.scene.input.MouseEvent
Direction	inout	
Description	When there is a double click in selected algorithms list, the selection is remove from algorithms selected.	
Stereotypes	FXML	

public handleSystemSaved (event : SystemSaved) : void

Parameters	event	
	Description	EventBus event.
	Multiplicity	1
	Type	 SystemSaved
	Direction	inout
Description	Handles the SystemSaved event, copying the path of system into input field.	

protected getRootPane () : javafx.scene.layout.Pane

Description	Root pane.
Return Type Description	Pane.

public algorithmsSelected (event : AlgorithmsSelectedEvent) : void

Parameters	event	
	Description	Event.
	Multiplicity	1
	Type	 AlgorithmsSelectedEvent
	Direction	inout
Description	Handles the AlgorithmsSelectedEvent published by the Eventbus. Adds all algorithms contained in the event to algorithms selected list.	

protected handleSelectInputDir (event : javafx.event.ActionEvent) : void

Parameters	event	
	Description	Event thrown when the File option of Select Input button is pressed.
	Multiplicity	1
	Type	javafx.event.ActionEvent
	Direction	inout
Description	Shows a file chooser for select the input system file.	
Stereotypes	FXML	

protected handleAddAlgAction (event : javafx.event.ActionEvent) : void

Parameters	event	
	Description	Event thrown when the <i>Add Algorithm</i> button is pressed.
	Multiplicity	1
	Type	javafx.event.ActionEvent
	Direction	inout
Description	When the <i>Add Algorithm</i> button is pressed, the <i>New Algorithm</i> window is shown.	
Stereotypes	FXML	

protected clear () : void

Description	Clears all fields.
-------------	--------------------

protected handleDeleteInputAction (event : javafx.event.ActionEvent) : void	
Parameters	event Description Event thrown when the contextual menu <i>Delete</i> option is pressed. Multiplicity 1 Type javafx.event.ActionEvent Direction inout
Description	Deletes the current selection in the inputs list.

NewBenchmarkModel

Name	Value
Description	New Benchmark view model.
Visibility	public

Attributes

private nameProperty : javafx.beans.property.StringProperty			
Description	Name binding property.		
Type	javafx.beans.property.StringProperty		
Getter	false	Setter	false
Multiplicity	1		

private algorithmsListProperty : javafx.beans.property.ListProperty			
Description	Algorithms list binding property.		
Type	 javafx.beans.property.ListProperty		
Getter	false	Setter	false
Multiplicity	1		

private algorithmsSelectedProperty : javafx.beans.property.ListProperty			
Description	Selected algorithms list binding property.		
Type	 javafx.beans.property.ListProperty		
Getter	false	Setter	true
Multiplicity	1		

private algorithmsFilteredList : javafx.collections.transformation.FilteredList			
Description	Algorithms filtered list.		
Stereotypes	Property		
Type	javafx.collections.transformation.FilteredList		
Getter	true	Setter	false
Multiplicity	1		

private inputDirProperty : javafx.beans.property.StringProperty			
Description	Input directory path binding property.		
Type	javafx.beans.property.StringProperty		
Getter	false	Setter	false
Multiplicity	1		

private inputFilesProperty : javafx.beans.property.ListProperty			
Description	Input files list binding property.		
Type	javafx.beans.property.ListProperty		
Getter	false	Setter	false
Multiplicity	1		

Operations

public NewBenchmarkModel ()	
Description	Constructor.

public nameProperty () : javafx.beans.property.StringProperty	
Description	Name.
Return Type Description	The name binding property.

public getName () : String	
Description	Gets the benchmark name from the nameProperty.
Return Type Description	Benchmark name.

public inputProperty () : javafx.beans.property.StringProperty	
Description	Input files directory binding property.
Return Type Description	Input files directory binding list property.

public getInputDir () : java.lang.String

Description	Gets the input files directory path from the inputDirProperty binding property.
Return Type Description	Input files directory absolute path.

public inputFilesListProperty () : javafx.beans.property.ListProperty

Description	Input files list binding property.
Return Type Description	Input files binding list property.

public getInputFilesList () : java.util.List

Description	Gets the input files list from the inputFilesListProperty binding property.
Return Type Description	Algorithms list.

public algorithmsListProperty () : javafx.beans.property.ListProperty

Description	Algorihtms list binding property.
Return Type Description	Algorithms list binding property.

public getAlgorithmsList () : java.util.List

Description	Gets the algorithms list from the algorithmsListProperty binding property.
Return Type Description	Current workspace registered algorithms.

public algorithmsSelectedProperty () : javafx.beans.property.ListProperty

Description	Algorithms list binding property.
Return Type Description	The algorithmsSelectedProperty.

public getAlgorithmsSelectedList () : java.util.List

Description	Gets the selected algorithms list from the algorithmsSelectedProperty binding property.
Return Type Description	Selected algorithms.

AlgorithmsClassLoadService

Name	Value
Description	Loads the algorithms found in the libraries included in the workspace lib folder.
Visibility	public

Attributes

private wsManager : WorkspaceManager			
Description	Workspace manager.		
Type	 WorkspaceManager		
Getter	false	Setter	false
Multiplicity	1		

Operations

public AlgorithmsLoadService ()	
Description	Constructor.
protected createTask () : javafx.concurrent.Task	
Description	Creates the background task which loads the found algorithms.
Return Type Description	Algorithms info list.
protected failed () : void	
Description	This method is executed when the background task is completed with errors.

BenchmarkSaveService

Name	Value
Description	Service which saves benchmark changes.
Visibility	public

Attributes

private benchmarksBean : BenchmarksBean			
Description	Benchmarks logic.		
Type	 BenchmarksBean		
Getter	false	Setter	false
Multiplicity	1		

private benchmark : Benchmark			
Description	Benchmark to save.		
Type	 Benchmark		
Getter	false	Setter	false
Multiplicity	1		

Operations

public BenchmarkSaveService (benchmark : Benchmark)		
Parameters	benchmark	
	Description	Benchmark to save.
	Multiplicity	1
	Type	 Benchmark
	Direction	inout
Description	Constructor.	

protected createTask () : javafx.concurrent.Task	
Description	Creates the background task which saves the benchmarks changes.
Return Type Description	Updated benchmark.

protected succeeded () : void	
Description	This method is executed when the background task is completed successfully. Publishes a BenchmarksChangeEvent and MessageEvent by the Eventbus.

protected failed () : void	
Description	This method is executed when the background task is completed with errors.

BenchmarksBean

Name	Value
Description	Business logic for insert, modify and delete algorithms.
Visibility	public

Attributes

private persistence : BenchmarksPersistence			
Description	Class for benchmarks persistence.		
Type	 BenchmarksPersistence		
Getter	false	Setter	false
Multiplicity	1		

Operations

public BenchmarksBean ()	
Description	Constructor.

protected setPersistence (persistence : BenchmarksPersistence) : void	
Parameters	persistence
	Multiplicity
	1
	Type
	 BenchmarksPersistence
	Direction
	inout
Description	For testing usage.

public update (benchmark : Benchmark) : void

Parameters	benchmark	
	Description	Benchmark.
	Multiplicity	1
	Type	 Benchmark
	Direction	inout
Description	Create the directory tree of benchmark.	
Exceptions	java.io.IOException	

public getBenchmarks (workspace : String) : java.util.List

Parameters	workspace	
	Description	Workspace path.
	Multiplicity	1
	Type	 String
	Direction	inout
Description	Returns the workspace registered benchmarks.	
Return Type Description	Registered benchmarks.	
Exceptions	java.lang.Exception	

public getBenchmark (name : String, workspace : String) : Benchmark

Parameters	name	
	Description	Benchmark name.
	Multiplicity	1
	Type	 String
	Direction	inout
Parameters	workspace	
	Description	Workspace path.
	Multiplicity	1
	Type	 String
	Direction	inout
Description	Returns a benchmark registered in a workspace, null if no exists.	

public getBenchmark (name : String, workspace : String) : Benchmark	
Return Type Description	Algorithm Entities list.
Exceptions	java.lang.Exception

public exists (name : String, workspace : String) : boolean																					
Parameters	<table border="1"> <thead> <tr> <th colspan="2">name</th> </tr> </thead> <tbody> <tr> <td>Description</td><td>Benchmark name.</td></tr> <tr> <td>Multiplicity</td><td>1</td></tr> <tr> <td>Type</td><td>● String</td></tr> <tr> <td>Direction</td><td>inout</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="2">workspace</th> </tr> </thead> <tbody> <tr> <td>Description</td><td>Workspace path.</td></tr> <tr> <td>Multiplicity</td><td>1</td></tr> <tr> <td>Type</td><td>● String</td></tr> <tr> <td>Direction</td><td>inout</td></tr> </tbody> </table>	name		Description	Benchmark name.	Multiplicity	1	Type	● String	Direction	inout	workspace		Description	Workspace path.	Multiplicity	1	Type	● String	Direction	inout
name																					
Description	Benchmark name.																				
Multiplicity	1																				
Type	● String																				
Direction	inout																				
workspace																					
Description	Workspace path.																				
Multiplicity	1																				
Type	● String																				
Direction	inout																				
Description	If exists a benchmark with the name argument in a workspace.																				
Return Type Description	true if exists a benchmark with the name argument, false otherwise.																				

■ BenchmarksPersistence

Name	Value
Description	Persist the benchmarks into an XML file entities using JAXB.
Visibility	public
Stereotypes	Singleton

Attributes

private me : BenchmarksPersistence			
Description	Single instance.		
Type	■ BenchmarksPersistence		
Getter	false	Setter	false
Multiplicity	1		

Operations

private BenchmarksPersistence ()

Description	Private constructor. For get a BenchmarksPersistence instance, will must be usage the static get() method.
-------------	--

public get () : BenchmarksPersistence

Description	Gets a single instance of BenchmarksPersistence.
Return Type Description	BenchmarksPersistence single instance.

public getBenchmarks (workspacePath : String) : Benchmarks

Parameters	workspacePath	
	Multiplicity	1
	Type	● String
	Direction	inout
Description	Returns the registered benchmarks in a workspace.	

public update (benchmark : Benchmark) : void

Parameters	benchmark	
	Description	Algorithms.
	Multiplicity	1
	Type	▀ Benchmark
	Direction	inout
Description	Initialize the benchmarks file with benchmarks parameter.	

public insert (algorithms : Algorithms) : void

Parameters	algorithms	
	Multiplicity	1
	Type	▀ Algorithms
	Direction	inout
Description	Add the algorithms of algorithms parameter to algorithms file.	

public insert (algorithm : AlgorithmInfo) : void	
Parameters	algorithm
	Description Algorithm.
	Multiplicity 1
	Type  AlgorithmInfo
	Direction inout
Description	Add an algorithm to algorithms file.

Benchmark

Name	Value
Description	Benchmark entity.
Visibility	public
Stereotypes	XmlRootElement

Attributes

private workspace : String			
Description	Workspace which the benchmark is registered.		
Stereotypes	Property, XmlAttribute		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		

private name : String			
Description	Benchmark Name.		
Stereotypes	Property, XmlAttribute		
Type	 String		
Getter	true	Setter	false
Multiplicity	1		

private inputsDir : String			
Description	Input implicational systems dir path.		
Stereotypes	Property, XmlAttribute		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		

private inputFiles : java.util.List			
Description	Input files which will be copied to input dir.		
Stereotypes	Property		
Type	java.util.List, XmlTransient		
Getter	true	Setter	true
Multiplicity	1..*		

private algorithmsEntities : java.util.List<AlgorithmInfo>			
Description	Benchmark algorithms.		
Stereotypes	Property, XmlElement		
Type	 AlgorithmInfo		
Getter	true	Setter	false
Multiplicity	*		

Operations

public Benchmark ()	
Description	Constructor.

public Benchmark (name : String, algorithms : java.util.List)	
Description	Constructor.
Exceptions	IllegalArgumentException if the name or algorithms list are empty.

public getBenchmarkPath () : String	
Description	The benchmark path.
Return Type Description	Benchmark path.

public getOutputDir () : String

Description	Path of output directory of benchmark.
Return Type Description	Output directory path.

public equals (obj : Object) : boolean

Parameters	obj
	Multiplicity
	1
	Type
	Object
	Direction
	inout
Description	Two benchmarks are equals if their names are.
Return Type Description	true if the obj parameter is a benchmarks and its name is equal to name of this benchmark instance, false otherwise.

public hashCode () : int

Description	Compute the hashcode based on the name property.
Return type description	Hashcode.

public toString () : String

Description	Benchmark string representation.
Return type description	Benchmark's name.

AlgorithmInfo

Name	Value
Description	Entity with an algorithm attributes.
Visibility	public
Stereotypes	XmlRootElement

Attributes

private name : String			
Description	Name.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		
Java Detail	N/A		

private shortName : String			
Description	Short name.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		

private type : Class			
Description	Algorithm implementation class.		
Stereotypes	Property		
Type	 java.lang.Class		
Getter	true	Setter	true
Multiplicity	1		

Operations

public AlgorithmInfo ()	
Description	Constructor.
Upper	1

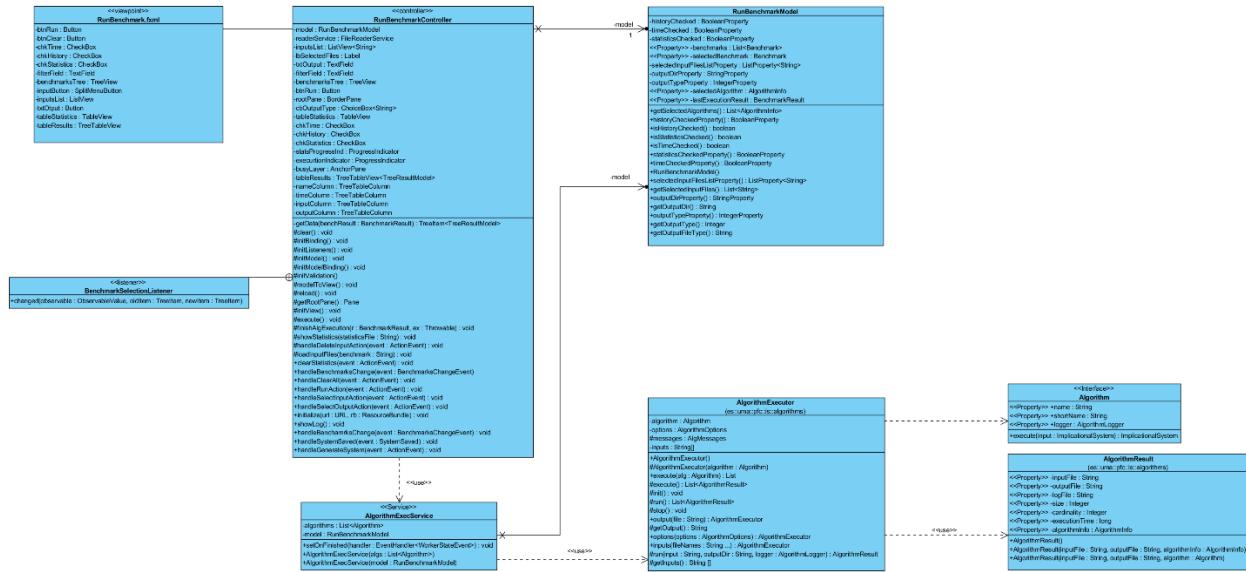
public AlgorithmInfo (alg : Algorithm)	
Parameters	alg Description Algorithm. Multiplicity 1 Type Algorithm Direction inout
Description	Constructs an AlgorithmInfo instance from an algorithm implementation.
Upper	1

public toString () : String	
Description	
Return type description	

public hashCode () : int	
Description	
Return type description	

public equals (obj : Object) : boolean	
Parameters	obj Multiplicity Unspecified Type Object Direction inout
Description	Two AlgorithmInfo objets are equals if their names an short names are.
Return type description	true if the obj parameter is a AlgorithmInfo instance and its name and short name are equal to name and short name of this benchmark instance, false otherwise.

6.5.5. Ejecutar Benchmarks



Summary

Name	Description
RunBenchmark.fxml	Benchmarks execution view.
RunBenchmarkController	Run view controller.
BenchmarkSelectionListener	Benchmarks tree listener.
RunBenchmarkModel	Benchmarks execution model.
AlgorithmExecService	Service for the background execution of algorithms and benchmarks.
AlgorithmExecutor	Service which executes an algorithm.
Algorithm	Algorithm of implicational system basis computation. It can be found in <i>API Algoritmos</i> class diagram.

Details

RunBenchmark.fxml

Name	Value
Description	Benchmarks execution view.
Visibility	public
Stereotypes	viewpoint

Attributes

private btnRun : javafx.scene.control.Button

Type	javafx.scene.control.Button		
Getter	false	Setter	false
Multiplicity	1		

private btnClear : javafx.scene.control.Button

Type	javafx.scene.control.Button		
Getter	false	Setter	false
Multiplicity	1		

private chkTime : javafx.scene.control.CheckBox

Description	Check for to enable / disable the Time mode.		
Type	javafx.scene.control.CheckBox		
Getter	false	Setter	false
Multiplicity	1		

private chkHistory : javafx.scene.control.CheckBox

Description	Check for to enable / disable the Trace mode.		
Type	javafx.scene.control.CheckBox		
Getter	false	Setter	false
Multiplicity	1		

private chkStatistics : javafx.scene.control.CheckBox

Description	Check for to enable / disable the Statistics mode.		
Type	javafx.scene.control.CheckBox		
Getter	false	Setter	false
Multiplicity	1		

private filterField : javafx.scene.control.TextField

Description	Filter over the benchmarks tree.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		

private benchmarksTree : javafx.scene.control.TreeView

Description	Two levels tree with the current workspace registered benchmarks at the first level, and its algorithms at the second level.		
Type	javafx.scene.control.TreeView		
Getter	false	Setter	false
Multiplicity	1		

private inputButton : javafx.scene.control.SplitMenuItem

Description	Menu button which shows the dialog box for to select the input files. Contains two options: select the existent file or generate a random implicational system.		
Type	javafx.scene.control.SplitMenuItem		
Getter	false	Setter	false
Multiplicity	1		

private inputsList : javafx.scene.control.ListView

Description	Input implicational system file paths list.		
Type	javafx.scene.control.ListView		
Getter	false	Setter	false
Multiplicity	1		
Leaf	false		

private txtOutput : javafx.scene.control.Button

Description	Text field for the output directory generated by the benchmark execution.		
Type	javafx.scene.control.Button		
Getter	false	Setter	false
Multiplicity	1		

private tableStatistics : javafx.scene.control.TableView			
Description	Table which contains the sizes statistics if the Statistics mode is enabled.		
Type	javafx.scene.control.TableView		
Getter	false	Setter	false
Multiplicity	1		

private tableResults : javafx.scene.control.TreeTableView			
Description	Table which contains the results of the executed benchmark, grouped by benchmark and algorithm.		
Type	javafx.scene.control.TreeTableView		
Getter	false	Setter	false
Multiplicity	1		

RunBenchmarkController

Name	Value		
Description	RunBenchmark view controller.		
Visibility	public		

Attributes

private model : RunBenchmarkModel			
Description	RunBenchmark view model.		
Type	 RunBenchmarkModel		
Getter	false	Setter	false
Multiplicity	1		

private readerService : FileReaderService			
Description	Service which reads a file in background.		
Type	 FileReaderService		
Getter	false	Setter	false
Multiplicity	1		

private inputsList : javafx.scene.control.ListView

Description	Input implicational system file paths list.		
Type	javafx.scene.control.ListView		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private lbSelectedFiles : javafx.scene.control.Label

Description	Input selected files selected count.		
Type	javafx.scene.control.Label		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private txtOutput : javafx.scene.control.TextField

Description	Output field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private filterField : javafx.scene.control.TextField

Description	Filter over benchmarks tree.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private benchmarksTree : javafx.scene.control.javafx.scene.control.TreeView

Description	Benchmarks and algorithms tree.		
Type	javafx.scene.control.javafx.scene.control.TreeView		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private btnRun : javafx.scene.control.Button			
Description	Run button.		
Type	javafx.scene.control.Button		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private rootPane : javafx.scene.layout.BorderPane			
Description	Root pane.		
Type	javafx.scene.layout.BorderPane		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private cbOutputType : javafx.scene.control.ChoiceBox			
Description	Output type dropdown.		
Type	javafx.scene.control.ChoiceBox		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private tableStatistics : javafx.scene.control.TableView			
Description	Table with statistics results.		
Type	javafx.scene.control.TableView		
Stereotypes	FXML		
Getter	false	Setter	false
Multiplicity	1		

private chkTime : javafx.scene.control.CheckBox			
Description	Time mode check.		
Type	javafx.scene.control.CheckBox		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private chkHistory : javafx.scene.control.CheckBox			
Description	Trace mode check.		
Type	javafx.scene.control.CheckBox		
Getter	false	Setter	false
Multiplicity	Unspecified		
Stereotypes	FXML		

private chkStatistics : javafx.scene.control.CheckBox			
Description	Statistics mode check.		
Type	javafx.scene.control.CheckBox		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private statsProgressInd : javafx.scene.control.ProgressIndicator			
Description	Statistics loading progress indicator.		
Type	javafx.scene.control.ProgressIndicator		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private executionIndicator : javafx.scene.control.ProgressIndicator			
Description	Execution progress indicator.		
Type	javafx.scene.control.ProgressIndicator		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private busyLayer : javafx.scene.layout.AnchorPane			
Description	Layer which contains progress indicators.		
Type	javafx.scene.layout.AnchorPane		
Getter	false	Setter	false
Multiplicity	1		

private tableResults : javafx.scene.control.TreeTableView			
Description	Execution results table.		
Type	javafx.scene.control.TreeTableView		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private nameColumn : javafx.scene.control.TreeTableColumn			
Description	Columns of execution results table.		
Type	javafx.scene.control.TreeTableColumn		
Getter	false	Setter	false
Multiplicity	1		

private timeColumn : javafx.scene.control.TreeTableColumn			
Description	Columns of execution results table.		
Type	javafx.scene.control.TreeTableColumn		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private inputColumn : javafx.scene.control.TreeTableColumn			
Description	Input file column of execution results table.		
Type	javafx.scene.control.TreeTableColumn		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private outputColumn : javafx.scene.control.TreeTableColumn			
Description	Columns of execution results table.		
Type	javafx.scene.control.TreeTableColumn		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

Operations

protected initModel () : void	
Description	Initializes the model loading the benchmarks into the tree.

protected initView () : void	
Description	Initializes the view.
Exceptions	IOException

protected initBinding () : void	
Description	Initializes the bindings between the components view and the model.

protected initListeners () : void	
Description	Initializes the components view listeners.

protected initModelBinding () : void	
Description	Initializes the bindings with the model.

public initialize (url : java.net.URL, rb : java.util.ResourceBundle) : void	
Parameters	url Description URL of the view. Multiplicity 1 Type java.net.URL Direction inout
	rb Description Resource bundle. Multiplicity 1 Type java.util.ResourceBundle Direction inout
Description	Initializes the controller.

protected modelToView () : void	
Description	The components that are not bound , will be updated with the model values.

private getData (benchResult : BenchmarkResult) : javafx.scene.control.TreeItem<javafx.scene.control.TreeItem>	
Parameters	benchResult Description Benchmark results. Multiplicity 1 Type  BenchmarkResult Direction inout
Description	Loads the benchmark results into a TreeTableView hierarchy.
Return Type Description	Benchmark result node.

protected reload () : void	
Description	Reloads the view and model.

protected getRootPane () : javafx.scene.layout.Pane	
Description	Returns the root pane.
Return Type Description	Root pane.

protected clear () : void

Description	Clears the model and the view.
-------------	--------------------------------

public clearStatistics (event : javafx.event.ActionEvent) : void

Parameters	event
	Description Event.
	Multiplicity 1
	Type javafx.event.ActionEvent
	Direction inout
Description	Clears the statistics table.

public showLog () : void

Description	Shows the algorithm result selected log.
-------------	--

protected execute () : void

Description	Executes the algorithms or benchmark selected with AlgorithmExecService.
-------------	--

public handleBenchmarksChange (event : BenchmarksChangeEvent) : void

Parameters	event
	Description Registered benchmarks change event.
	Multiplicity 1
	Type  BenchmarksChangeEvent
	Direction inout
Description	Handles the event BenchmarksChangeEvent published by Eventbus, and reloads the view and the model.
Stereotypes	FXML

public handleClearAll (event : javafx.event.ActionEvent) : void

Parameters	event	
	Description	Event.
	Multiplicity	1
	Type	javafx.event.ActionEvent
	Direction	inout
Description	ActionEvent handler of Run button. Clear all traces.	
Stereotypes	FXML	

public handleRunAction (event : javafx.event.ActionEvent) : void

Parameters	event	
	Description	Event.
	Multiplicity	1
	Type	javafx.event.ActionEvent
	Direction	inout
Description	Handles the event thrown when Run button is pressed. Runs the selected benchmark / algorithm.	
Stereotypes	FXML	

public handleSelectInputAction (event : javafx.event.ActionEvent) : void

Parameters	event	
	Description	Event.
	Multiplicity	1
	Type	javafx.event.ActionEvent
	Direction	inout
Description	Shows a dialog box for select the input of algorithm.	

public handleSystemSaved (event : SystemSaved) : void	
Parameters	event Description Event. Multiplicity 1 Type SystemSaved Direction inout
Description	Handles the SystemSaved event, copying the path of system into input field.

public handleGenerateSystem (event : javafx.event.ActionEvent) : void	
Parameters	event Description Action event. Multiplicity 1 Type javafx.event.ActionEvent Direction inout
Description	Shows the generator window for generate a random system.
Stereotypes	FXML

public handleSelectOutputAction (event : javafx.event.ActionEvent) : void	
Parameters	event Description Event. Multiplicity 1 Type javafx.event.ActionEvent Direction inout
Description	Shows the dialog box for select the target of algorithm results.
Stereotypes	FXML

public handleBenchamrksChange (event : BenchmarksChangeEvent) : void	
Parameters	
	event
Description	Event.
Multiplicity	1
Type	BenchmarksChangeEvent
Direction	inout
Description	Handles the event BenchmarksChangeEvent published by Eventbus, and reloads view and model.

protected loadInputFiles (benchmark : String) : void	
Parameters	
	benchmark
Description	Benchmark's name.
Multiplicity	1
Type	String
Direction	inout
Description	Loads the input files defined for the selected benchmark.

protected handleDeleteInputAction (event : javafx.event.ActionEvent) : void	
Parameters	
	event
Description	Event.
Multiplicity	1
Type	javafx.event.ActionEvent
Direction	inout
Description	Removes the selected items from the list.
Stereotypes	FXML

protected finishAlgExecution (result : BenchmarkResult, ex : Throwable) : void

Parameters	result	
	Description	Benchmark result.
	Multiplicity	1
	Type	 BenchmarkResult
	Direction	inout
	ex	
	Description	Exception if the execution is failed. Null otherwise.
	Multiplicity	1
	Type	Throwable
	Direction	inout
Description	Shows the results and statistics.	

protected showStatistics (statisticsFile : String) : void

Parameters	statisticsFile	
	Description	Statistics file path.
	Multiplicity	1
	Type	 String
	Direction	inout
Description	Loads the result statistics into a table.	

 **RunBenchmarkModel**

Name	Value
Description	Benchmarks execution model.
Visibility	public

Attributes

private historyChecked : javafx.beans.property.BooleanProperty			
Description	Trace mode checked binding property.		
Type	javafx.beans.property.BooleanProperty		
Getter	false	Setter	false
Multiplicity	1		

private timeChecked : javafx.beans.property.BooleanProperty			
Description	Time mode checked binding property.		
Type	javafx.beans.property.BooleanProperty		
Getter	false	Setter	false
Multiplicity	1		

private statisticsChecked : javafx.beans.property.BooleanProperty			
Description	Statistics mode checked binding property.		
Type	javafx.beans.property.BooleanProperty		
Getter	false	Setter	false
Multiplicity	1		

private benchmarks : java.util.List<Benchmark>			
Description	Available benchmarks.		
Stereotypes	Property		
Type	java.util.List<Benchmark>		
Getter	true	Setter	true
Multiplicity	0..*		

private selectedBenchmark : Benchmark			
Description	Selected benchmark.		
Stereotypes	Property		
Type	 Benchmark		
Getter	true	Setter	true
Multiplicity	1		

private selectedInputFilesListProperty : javafx.beans.property.ListProperty			
Description	Input files list binding property.		
Type	javafx.beans.property.ListProperty		
Getter	false	Setter	false
Multiplicity	1		

private outputDirProperty : javafx.beans.property.StringProperty			
Description	Output file path binding property.		
Type	javafx.beans.property.StringProperty		
Getter	false	Setter	false
Multiplicity	1		

private outputTypeProperty : javafx.beans.property.IntegerProperty			
Description	Output file type selected binding property.		
Type	javafx.beans.property.IntegerProperty		
Getter	false	Setter	false
Multiplicity	1		

private selectedAlgorithm : AlgorithmInfo			
Description	When the selection is an algorithm, the selected algorithm info.		
Stereotypes	Property		
Type	 AlgorithmInfo		
Getter	true	Setter	true
Multiplicity	1		

private lastExecutionResult : BenchmarkResult			
Description	Last execution result.		
Stereotypes	Property		
Type	 BenchmarkResult		
Getter	true	Setter	true
Multiplicity	1		

Operations

public RunBenchmarkModel ()

Description	Constructor.
-------------	--------------

public getSelectedAlgorithms () : java.util.List<AlgorithmInfo>

Description	Selected algorithms.
-------------	----------------------

Return Type Description	List of selected algorithms info.
-------------------------	-----------------------------------

public historyCheckedProperty () : javafx.beans.property.BooleanProperty

Description	Trace checked binding property.
-------------	---------------------------------

Return Type Description	The trace checked binding property.
-------------------------	-------------------------------------

public isHistoryChecked () : boolean

Description	If the trace mode is checked.
-------------	-------------------------------

Return Type Description	true if the history mode is checked, false otherwise.
-------------------------	---

public isStatisticsChecked () : boolean

Description	If the statistics mode is checked.
-------------	------------------------------------

Return Type Description	true if the statistics mode is checked, false otherwise.
-------------------------	--

public isTimeChecked () : boolean

Description	If the time mode is checked.
-------------	------------------------------

Return Type Description	true if the Time mode is checked, false otherwise.
-------------------------	--

public statisticsCheckedProperty () : javafx.beans.property.BooleanProperty

Description	Statistics checked binding property.
-------------	--------------------------------------

Return Type Description	The statistics checked binding property.
-------------------------	--

public timeCheckedProperty () : javafx.beans.property.BooleanProperty

Description	Time checked binding property.
-------------	--------------------------------

Return Type Description	The time checked binding property.
-------------------------	------------------------------------

public selectedInputFilesListProperty () : javafx.beans.property.ListProperty

Description	Path input files binding property.
Return Type Description	The Path input files binding property.

public getSelectedInputFiles () : java.util.List<String>

Description	Gets the selected input files paths from the selectedInputFilesListProperty.
Return Type Description	The selected input files paths.

public outputDirProperty () : javafx.beans.property.StringProperty

Description	Path output directory binding property.
Return Type Description	The Path output directory property.

public getOutputDir () : String

Description	Returns the output directory path.
Return Type Description	The output directory path.

public outputTypeProperty () : javafx.beans.property.IntegerProperty

Description	Output type binding property.
Return Type Description	The Output type binding property.

public getOutputType () : Integer

Description	Gets the output file type from the outputTypeProperty.
Return Type Description	The output type.

public getOutputFileType () : String

Description	Returns the file type of the selected output type.
Return Type Description	File Type.



RunBenchmarkController.BenchmarkSelectionListener

Name	Value
Description	Benchmarks tree listener.
Visibility	protected
Stereotypes	listener

Operations

public changed (observable : ObservableValue, oldItem : javafx.scene.control.TreeItem, newItem : javafx.scene.control.TreeItem)																						
Parameters	<table border="1"><tr><td>observable</td></tr><tr><td>Multiplicity</td><td>1</td></tr><tr><td>Type</td><td>ObservableValue</td></tr><tr><td>Direction</td><td>inout</td></tr></table> <table border="1"><tr><td>oldItem</td></tr><tr><td>Multiplicity</td><td>1</td></tr><tr><td>Type</td><td>javafx.scene.control.TreeItem</td></tr><tr><td>Direction</td><td>inout</td></tr></table> <table border="1"><tr><td>newItem</td></tr><tr><td>Multiplicity</td><td>1</td></tr><tr><td>Type</td><td>javafx.scene.control.TreeItem</td></tr><tr><td>Direction</td><td>inout</td></tr></table>	observable	Multiplicity	1	Type	ObservableValue	Direction	inout	oldItem	Multiplicity	1	Type	javafx.scene.control.TreeItem	Direction	inout	newItem	Multiplicity	1	Type	javafx.scene.control.TreeItem	Direction	inout
observable																						
Multiplicity	1																					
Type	ObservableValue																					
Direction	inout																					
oldItem																						
Multiplicity	1																					
Type	javafx.scene.control.TreeItem																					
Direction	inout																					
newItem																						
Multiplicity	1																					
Type	javafx.scene.control.TreeItem																					
Direction	inout																					
Description	<p>This method is invoked when the selection on the benchmarks tree is changed.</p> <ul style="list-style-type: none">When a benchmark is selected, the output field is initialized with the benchmark output directory and is disabled.In this case, the mode checks and console outputs are disabled too.If the selection is an algorithm, the output field is enabled and initialized with the path of algorithm default output file.If the selection is more than one algorithm, the output field is cleared and disabled.If the selection contains benchmarks and algorithms the output field is cleared and the Run button disabled.																					



AlgorithmExecService

Name	Value
Description	Service for the background execution of algorithms and benchmarks.
Visibility	public

Attributes

private algorithms : java.util.List<Algorithm>			
Description	Algorithms to execute.		
Type	java.utilList<Algorithm>		
Getter	false	Setter	false
Multiplicity	1..*		

private model : RunBenchmarkModel			
Description	Model.		
Type	RunBenchmarkModel		
Getter	false	Setter	false
Multiplicity	1		

Operations

protected createTask () : javafx.concurrent.Task	
Description	Task for execute algorithms and benchmarks. If the Statistics mode is enabled, prints the results into a CSV file.

protected instanceAlgorithms () : void	
Description	Creates the instances of the benchmark algorithms.

protected getOptions () : AlgorithmOptions	
Description	Establishes the algorithm options from the model.
Return Type Description	Algorithm execution options.

public setOnFinished (handler : javafx.event.EventHandler) : void	
Parameters	handler Description Handler. Multiplicity 1 Type javafx.event.EventHandler Direction inout
Description	The onFinish event handler is called whenever the Task state transitions to the finished state: CANCELLED, FAILED or SUCCEEDED.

protected failed () : void	
Description	This method is executed when the background task is completed with errors.

protected succeeded () : void	
Description	Publishes a message event when the task is finished successfully.

public AlgorithmExecService (algs : java.util.List<Algorithm>)	
Parameters	algs Description Algorithms to execute. Multiplicity 1 Type java.util.List<Algorithm> Direction inout
Description	Constructor.

public AlgorithmExecService (model : RunBenchmarkModel)	
Parameters	model Description Model. Multiplicity 1 Type RunBenchmarkModel Direction inout
Description	Constructor.

protected instanceAlgorithm (algorithm : AlgorithmInfo) : Algorithm	
Parameters	
	algorithm
Description	Algorithm info.
Multiplicity	1
Type	 AlgorithmInfo
Direction	inout
Description	Instances the algorithm of type of algorithm parameter.
Return Type Description	Algorithm instance.

protected printResults (result : BenchmarkResult) : void	
Parameters	
	result
Description	Results.
Multiplicity	1
Type	 BenchmarkResult
Direction	inout
Description	Prints the results into a CSV file.
Exceptions	IOException when a read /write error occur.

AlgorithmExecutor

Name	Value
Description	Service which executes an algorithm.
Visibility	public

Attributes

private algorithm : Algorithm			
Description	Algorithm to execute.		
Type	 Algorithm		
Getter	false	Setter	false
Multiplicity	1		

private options : AlgorithmOptions			
Description	Execution options.		
Type	 AlgorithmOptions		
Getter	false	Setter	false
Multiplicity	1		

protected messages : AlgMessages			
Description	I18n messages.		
Type	 AlgMessages		
Getter	false	Setter	false
Multiplicity	1		

private inputs : String			
Description	Paths of inputs system.		
Type	 String		
Getter	false	Setter	false
Multiplicity	0..*		

Operations

public AlgorithmExecutor ()	
Description	Constructor.

protected AlgorithmExecutor (algorithm : Algorithm)	
Parameters	algorithm
	Description Algorithm to execute.
	Multiplicity Unspecified
	Type  Algorithm
	Direction inout
Description	Protected constructor. For testing purpose only.

public execute (alg : Algorithm) : java.util.List

Parameters	alg	
	Description	Algorithm.
	Multiplicity	1
	Type	 Algorithm
	Direction	inout
Description	Executes an algorithm.	
Return Type Description	Algorithm results.	

protected execute () : java.util.List<AlgorithmResult>

Description	Executes an algorithm in three stages: initialization, execution and finalization.
Return Type Description	Execution results.

protected init () : void

Description	Initializes the algorithm execution.
-------------	--------------------------------------

protected run () : java.util.List<AlgorithmResult>

Description	Executes the algorithm with the inputs and options established.
Return Type Description	Algorithm results.
Exceptions	<ul style="list-style-type: none">java.io.IOException if IO error occurs.java.lang.IllegalArgumentException if the algorithm is null.

public output (file : String) : AlgorithmExecutor

Parameters	file	
	Description	Path of the output.
	Multiplicity	Unspecified
	Type	String
	Direction	inout
Description	Sets the output path of the result execution.	
Return Type Description	AlgorithmExecutor with output system established.	

protected getOutput () : String

Description	Returns the output path.
Return Type Description	Output path.

public options (options : AlgorithmOptions) : AlgorithmExecutor

Parameters	options	
	Description	Execution options.
	Multiplicity	Unspecified
	Type	 AlgorithmOptions
	Direction	inout
Description	Sets an execution options.	
Return Type Description	AlgorithmExecutor with an execution option established.	

public inputs (fileNames : String) : AlgorithmExecutor

Parameters	fileNames	
	Description	Additional inputs.
	Multiplicity	0..*
	Type Modifier	...
	Type	 String
	Direction	inout
Description	Sets the path of the inputs system.	
Return Type Description	AlgorithmExecutor with inputs system setted.	

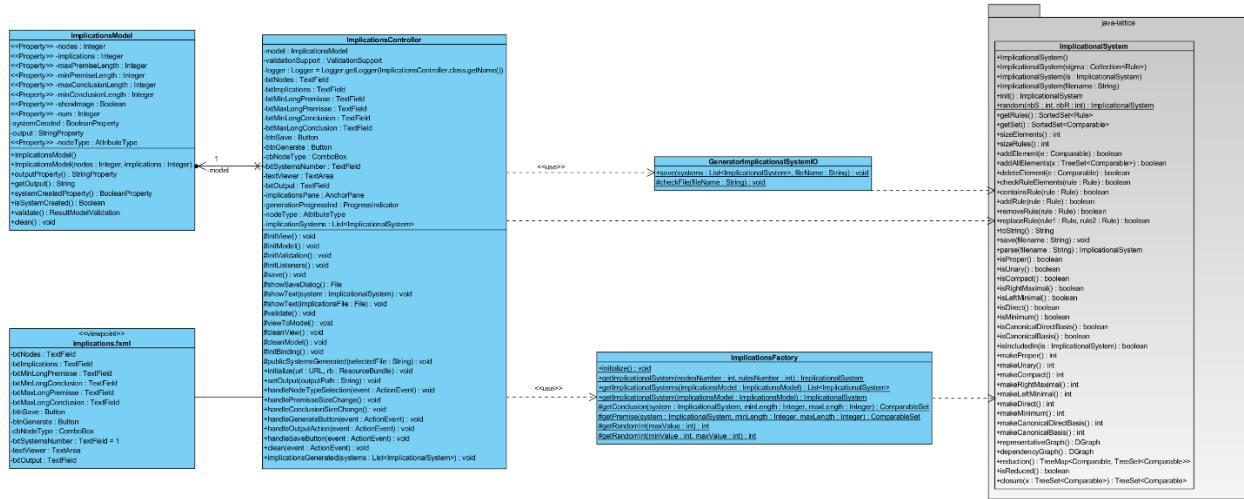
protected getInputs () : String

Description	Returns the inputs path.
Return Type Description	Inputs path.

protected run (input : String, outputDir : String, logger : AlgorithmLogger) : AlgorithmResult

input	
Description	Input implicational system.
Multiplicity	Unspecified
Type	● String
Direction	inout
outputDir	
Description	Output dir.
Multiplicity	Unspecified
Type	● String
Direction	inout
logger	
Description	Logger.
Multiplicity	Unspecified
Type	■ AlgorithmLogger
Direction	inout
Description	Executes the algorithm with an input and output dir.
Return Type Description	Algorithm result.

6.5.6. Generador Implicaciones



Summary

Name	Description
implications.fxml	Implications generator main view.
ImplicationsController	Implications generator view's controller.
ImplicationsModel	Class which represents the features of implicational system to generate.
fr.kbertet.lattice.ImplicationalSystem	This class gives a representation for an implicational system (fr.kbertet.lattice.ImplicationalSystem), a set of rules. It belongs to the java-lattice library.
ImplicationsFactory	Implicational systems factory.
GeneratorImplicationalSystemIO	Implements the methods to save implicational systems in files.

Details



implications.fxml

Name	Value
Description	Implications generator main view.
Visibility	public
Stereotypes	viewpoint

Attributes

private txtNodes : TextField

Description	System nodes number field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		

private txtImplications : javafx.scene.control.TextField

Description	Implications number field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		

private txtMinLongPremisse : javafx.scene.control.TextField

Description	Premises min length field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		

private txtMinLongConclusion : javafx.scene.control.TextField

Description	Conclusions min length field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		

private txtMaxLongPremisse : javafx.scene.control.TextField			
Description	Premises max length field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		

private txtMaxLongConclusion : javafx.scene.control.TextField			
Description	Conclusions max length field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		

private btnSave : javafx.scene.control.Button			
Description	Save button.		
Type	javafx.scene.control.Button		
Getter	false	Setter	false
Multiplicity	1		

private btnGenerate : javafx.scene.control.Button			
Description	Generate button.		
Type	javafx.scene.control.Button		
Getter	false	Setter	false
Multiplicity	1		

private cbNodeType : javafx.scene.control.ComboBox			
Description	Dropdown with the node types: numerics (0, 1,, 2, ...), alphabeticals (a, b, c, ...) or alphanumerics (a0, a1, a2, ...).		
Type	javafx.scene.control.ComboBox		
Getter	false	Setter	false
Multiplicity	1		

private txtSystemsNumber : javafx.scene.control.TextField			
Description	Systems number field.		
Initial Value	1		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		

private textViewer : javafx.scene.control.TextArea			
Description	Viewer.		
Type	javafx.scene.controlTextArea		
Getter	false	Setter	false
Multiplicity	1		

private txtOutput : javafx.scene.control.TextField			
Description	Path output file field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		



ImplicationsController

Name	Value
Description	Implications generator view's controller.
Visibility	public

Attributes

private model : ImplicationsModel			
Description	Model.		
Type	ImplicationsModel		
Getter	false	Setter	false
Multiplicity	1		

private validationSupport : org.controlsfx.validation.ValidationSupport			
Description	Validation support.		
Type	org.controlsfx.validation.ValidationSupport		
Getter	false	Setter	false
Multiplicity	1		

private logger : java.util.logging.Logger			
Initial Value	Logger.getLogger(ImplicationsController.class.getName())		
Type	java.util.logging.Logger		
Getter	false	Setter	false
Multiplicity	1		

private txtNodes : javafx.scene.control.TextField			
Description	Nodes number field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private txtImplications : javafx.scene.control.TextField			
Description	Implications number field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		

private txtMinLongPremisse : javafx.scene.control.TextField			
Description	Min length premise field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private txtMaxLongPremisse : javafx.scene.control.TextField			
Description	Max length premise field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		

private txtMinLongConclusion : javafx.scene.control.TextField			
Description	Min length conclusion field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private txtMaxLongConclusion : javafx.scene.control.TextField			
Description	Max length conclusion field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private btnSave : javafx.scene.control.javafx.scene.control.Button			
Description	Save button.		
Type	javafx.scene.control.javafx.scene.control.Button		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private btnGenerate : javafx.scene.control.javafx.scene.control.Button			
Description	Generate button.		
Type	javafx.scene.control.javafx.scene.control.Button		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private cbNodeType : javafx.scene.control.ComboBox			
Description	Node types dropdown.		
Type	javafx.scene.control.ComboBox		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private txtSystemsNumber : javafx.scene.control.TextField			
Description	Number systems to generate field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private textViewer : javafx.scene.control.TextArea			
Description	Generated systems viewer.		
Type	javafx.scene.control.TextArea		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private txtOutput : javafx.scene.control.TextField			
Description	Output file path field.		
Type	javafx.scene.control.TextField		
Getter	false	Setter	false
Stereotypes	FXML		

private implicationsPane : javafx.scene.layout.AnchorPane			
Description	Root pane.		
Type	javafx.scene.layout.AnchorPane		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private generationProgressInd : javafx.scene.control.ProgressIndicator			
Description	System generation progress indicator.		
Type	javafx.scene.control.ProgressIndicator		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private nodeType : AttributeType			
Description	Selected node type.		
Type	 AttributeType		
Getter	false	Setter	false
Multiplicity	1		

private implicationSystems : java.util.List<fr.kbertet.lattice.ImplicationalSystem>			
Description	Generated system.		
Type	java.util.List<fr.kbertet.lattice.ImplicationalSystem>		
Getter	false	Setter	false
Derived	false		
Multiplicity	0..*		

Operations

public initialize (url : java.net.URL, rb : java.util.ResourceBundle) : void			
Parameters	url		
	Description	URL of the view.	
	Multiplicity	1	
	Type	java.net.URL	
	Direction	inout	
	rb		
	Description	Resource Bundle.	
	Multiplicity	1	
	Type	java.util.ResourceBundle	
	Direction	inout	
Description	Initializes the controller class.		

protected initView () : void

Description	Initializes the view.
-------------	-----------------------

protected initModel () : void

Description	Creates an instance of ImplicationsModel.
-------------	---

protected viewToModel () : void

Description	Reads the view values and saves them into the model.
-------------	--

protected initBindingjav() : void

Description	Initializes the bindings.
-------------	---------------------------

protected initValidation () : void

Description	Initializes the validation support.
-------------	-------------------------------------

protected initListeners () : void

Description	Initializes the listeners.
-------------	----------------------------

public setOutput (outputPath : String) : void

Parameters	outputPath	
	Description	Output file path.
	Multiplicity	1
	Type	String
	Direction	inout
Description	Sets the output file path into the output field.	

public handlePremisseSizeChange () : void

Description	Cleans the max and min premise length fields decorators. This method is invoked by the max and min premise fields listeners.
-------------	---

public handleConclusionSizeChange () : void

Description	Cleans the max and min conclusion length fields' decorators. This method is invoked by the max and min conclusion fields listeners.
-------------	--

public handleNodeTypeSelection (event : javafx.event.ActionEvent) : void

Parameters	event
	Description
	Multiplicity
	Type
	Direction
Description	When a node type is selected from the node types dropdown, the node type field is updated.
Stereotypes	FXML

public handleOutputAction (event : javafx.event.ActionEvent) : void

Parameters	event
	Description
	Multiplicity
	Type
	Direction
Description	When the <i>Search (...)</i> button is pressed, opens the dialog box to select an output file, and copy the selected file path into Output field.
Stereotypes	FXML

public clean (event : javafx.event.ActionEvent) : void											
Parameters	<table border="1"> <tr> <td colspan="2">event</td></tr> <tr> <td>Description</td><td>Event thrown when <i>Clean</i> button is pressed.</td></tr> <tr> <td>Multiplicity</td><td>1</td></tr> <tr> <td>Type</td><td>javafx.event.ActionEvent</td></tr> <tr> <td>Direction</td><td>inout</td></tr> </table>	event		Description	Event thrown when <i>Clean</i> button is pressed.	Multiplicity	1	Type	javafx.event.ActionEvent	Direction	inout
event											
Description	Event thrown when <i>Clean</i> button is pressed.										
Multiplicity	1										
Type	javafx.event.ActionEvent										
Direction	inout										
Description	Handles the event thrown when the Clean button is pressed and cleans the form fields.										
Stereotypes	FXML										

public handleGenerateButton (event : javafx.event.ActionEvent) : void											
Parameters	<table border="1"> <tr> <td colspan="2">event</td></tr> <tr> <td>Description</td><td>Event.</td></tr> <tr> <td>Multiplicity</td><td>1</td></tr> <tr> <td>Type</td><td>javafx.event.ActionEvent</td></tr> <tr> <td>Direction</td><td>inout</td></tr> </table>	event		Description	Event.	Multiplicity	1	Type	javafx.event.ActionEvent	Direction	inout
event											
Description	Event.										
Multiplicity	1										
Type	javafx.event.ActionEvent										
Direction	inout										
Description	Handles the event thrown when the <i>Generate</i> button is pressed and generates the implications with the inserted values.										
Stereotypes	FXML										

public handleSaveButton (event : javafx.event.ActionEvent) : void											
Parameters	<table border="1"> <tr> <td colspan="2">event</td></tr> <tr> <td>Description</td><td>Even thrown when the <i>Save</i> button is pressed.</td></tr> <tr> <td>Multiplicity</td><td>1</td></tr> <tr> <td>Type</td><td>javafx.event.ActionEvent</td></tr> <tr> <td>Direction</td><td>inout</td></tr> </table>	event		Description	Even thrown when the <i>Save</i> button is pressed.	Multiplicity	1	Type	javafx.event.ActionEvent	Direction	inout
event											
Description	Even thrown when the <i>Save</i> button is pressed.										
Multiplicity	1										
Type	javafx.event.ActionEvent										
Direction	inout										
Description	Saves the implicational system in the path inserted into Output filed.										
Stereotypes	FXML										

protected cleanView () : void

Description	Cleans the view fields.
-------------	-------------------------

protected cleanModel () : void

Description	Cleans the model values.
-------------	--------------------------

protected validate () : void

Description	Performs the model validation.
-------------	--------------------------------

Exceptions	RuntimeException If validation error exists.
------------	--

protected save () : void

Description	Saves the generated implicational systems with the GeneratorImplicationalSystemIO class.
-------------	--

protected showSaveDialog () : java.io.File

Description	Shows the <i>Save</i> dialog box and returns the selected file.
-------------	---

Return Type Description	Selected file.
-------------------------	----------------

protected showText (system : fr.kbertet.lattice.ImplicationalSystem) : void

Parameters	system	
	Description	Implicational system.
	Multiplicity	1
	Type	 fr.kbertet.lattice.ImplicationalSystem
	Direction	inout
Description	Shows in the viewer an implicational system.	

protected showText (implicationsFile : java.io.File) : void	
Parameters	implicationsFile
	Description Implicational system.
	Multiplicity 1
	Type java.io.File
	Direction inout
Description	Shows in the viewer a saved system.

public implicationsGenerated (systems : java.util.List<Implications>) : void	
Parameters	systems
	Description Generated implications.
	Multiplicity 1
	Type java.util.List<Implications>
	Direction inout
Description	Shows the generated system in the viewer, if was generated only one. If was generated several, shows the <i>Save</i> dialog box.

protected publicSystemsGenerated (selectedFile : String) : void	
Parameters	selectedFile
	Description Output file path.
	Multiplicity 1
	Type  String
	Direction inout
Description	Publishes a SystemSaved event by Eventbus.

ImplicationsModel

Name	Value
Description	Class which represents the features of implicational system to generate.
Visibility	public

Attributes

private nodes : Integer			
Description	Nodes number.		
Stereotypes	Property		
Type	Integer		
Getter	true	Setter	true
Multiplicity	1		

private implications : Integer			
Description	Implications number.		
Stereotypes	Property		
Type	Integer		
Getter	true	Setter	true
Multiplicity	1		

private maxPremiseLength : Integer			
Description	Premise attributes max length.		
Stereotypes	Property		
Type	Integer		
Getter	true	Setter	true
Multiplicity	1		

private minPremiseLength : Integer			
Description	Premise attributes min length.		
Stereotypes	Property		
Type	Integer		
Getter	true	Setter	true
Multiplicity	1		

private maxConclusionLength : Integer			
Description	Conclusion attributes max length.		
Stereotypes	Property		
Type	Integer		
Getter	true	Setter	true
Multiplicity	1		

private minConclusionLength : Integer			
Description	Conclusion attributes min length.		
Stereotypes	Property		
Type	Integer		
Getter	true	Setter	true
Multiplicity	1		

private showImage : Boolean			
Description	If the graph will be painted.		
Stereotypes	Property		
Type	Boolean		
Getter	true	Setter	true
Multiplicity	1		

private num : Integer			
Description	Number of systems to generate.		
Stereotypes	Property		
Type	Integer		
Getter	true	Setter	true
Multiplicity	1		

private systemCreated : javafx.beans.property.BooleanProperty			
Description	System created flag binding property.		
Type	javafx.beans.property.BooleanProperty		
Getter	false	Setter	false
Multiplicity	1		

private output : javafx.beans.property.StringProperty			
Description	Output file path binding property.		
Type	javafx.beans.property.StringProperty		
Getter	false	Setter	false
Multiplicity	1		

private nodeType : AttributeType			
Description	Node types. It can be numeric, alphabetical or alphanumeric.		
Stereotypes	Property		
Type	 AttributeType		
Getter	true	Setter	true
Multiplicity	1		

Operations

public ImplicationsModel ()	
Description	Constructor.

public ImplicationsModel (nodes : Integer, implications : Integer)	
	nodes
Description	Nodes number
Multiplicity	1
Type	Integer
Direction	inout
	implications
Description	Implications number.
Multiplicity	1
Type	Integer
Direction	inout
Description	Constructor.

public outputProperty () : javafx.beans.property.StringProperty	
Description	Returns the output file path binding property.
Return Type Description	The output file path binding property.

public getOutput () : String

Description	Gets the output file path from the outputProperty property.
Return Type Description	Ouptut file path.

public systemCreatedProperty () : javafx.beans.property.BooleanProperty

Description	Returns the system created flag binding property.
Return Type Description	The system created flag binding property.

public isSystemCreated () : Boolean

Description	Gets if the system has been created from the systemCreatedProperty property.
Return Type Description	true if the system has been created, false otherwise.

public validate () : ResultModelValidation

Description	Checks if the model is correct for the implicational system generation.
Return Type Description	Object with the validation results.

public clean () : void

Description	Cleans the properties values.
-------------	-------------------------------

**ImplicationsFactory**

Name	Value
Description	Random implicational systems factory.
Visibility	public

Operations**public initialize () : void**

Description	Initializes the factory.
-------------	--------------------------

public getImplicationalSystems (implicationsModel : ImplicationsModel) : java.util.List

Parameters	implicationsModel	
	Description	Restrictions of the systems to generate.
	Multiplicity	1
	Type	 ImplicationsModel
	Direction	inout
Description	Returns n implicational systems fulfilling the established restrictions in the implicationsModel parameter.	
Return Type Description	Implicational systems list.	

public getImplicationalSystem (nodesNumber : int, rulesNumber : int) : fr.kbertet.lattice.ImplicationalSystem

Parameters	nodesNumber	
	Description	Nodes number.
	Multiplicity	1
	Type	 int
	Direction	inout
	rulesNumber	
	Description	Implications number.
	Multiplicity	1
	Type	 int
	Direction	inout
Description	Generates a random system with the nodes and implications number passed by parameter.	
Return Type Description	Generated implicational system.	

<u>public getImplicationalSystem (implicationsModel : ImplicationsModel) :</u>
<u>fr.kbertet.lattice.ImplicationalSystem</u>

Parameters	implicationsModel	
	Description	Restrictions of the systems to generate.
	Multiplicity	1
	Type	 ImplicationsModel
Description	Direction	
	inout	
Return Type Description		Random implicational system.

<u>protected getRandomInt (minValue : int, maxValue : int) : int</u>

Parameters	minValue	
	Description	Min value.
	Multiplicity	1
	Type	 int
	Direction	inout
	maxValue	
	Description	Max value.
	Multiplicity	1
Description	Type	 int
	Direction	inout
Return Type Description		Random integer value.

protected getRandomInt (maxValue : int) : int

Parameters	maxValue	
	Description	Max value.
	Multiplicity	1
	Type	int
	Direction	inout
Description	Returns a random integer, between 0 and maxValue parameter value.	

protected getConclusion (system : fr.kbertet.lattice.ImplicationalSystem, minLength : Integer, maxLength : Integer) : ComparableSet

Parameters	system	
	Description	Implicational system.
	Multiplicity	1
	Type	fr.kbertet.lattice.ImplicationalSystem
	Direction	inout
	minLength	
	Description	Min nodes number.
	Multiplicity	1
	Type	Integer
	Direction	inout
	maxLength	
	Description	Max nodes number.
	Multiplicity	1
	Type	Integer
	Direction	inout
Description	Returns a conclusion with a max nodes number. If maxLength parameter is null or less than 0, the max nodes number is the system nodes number.	
Return Type Description	Conclusion with a max nodes number.	
Exceptions	RuntimeException if the maxLength parameter is greater than attributes number.	

protected getPremise (system : fr.kbertet.lattice.ImplicationalSystem, minLength : Integer, maxLength : Integer) : ComparableSet

Parameters	system	
	Description	Implicational system.
	Multiplicity	1
	Type	 fr.kbertet.lattice.Implicational System
	Direction	inout
	minLength	
	Description	Min nodes number.
	Multiplicity	1
	Type	Integer
	Direction	inout
	maxLength	
	Description	Max nodes number.
	Multiplicity	1
	Type	Integer
	Direction	inout
Description	Returns a premise with a max nodes number. If the maxLength parameter is null or less than 0, the max nodes number is the system nodes number.	
Return Type Description	Premise with a max nodes number.	
Return Type Description	Random integer value.	

GeneratorImplicationalSystemIO

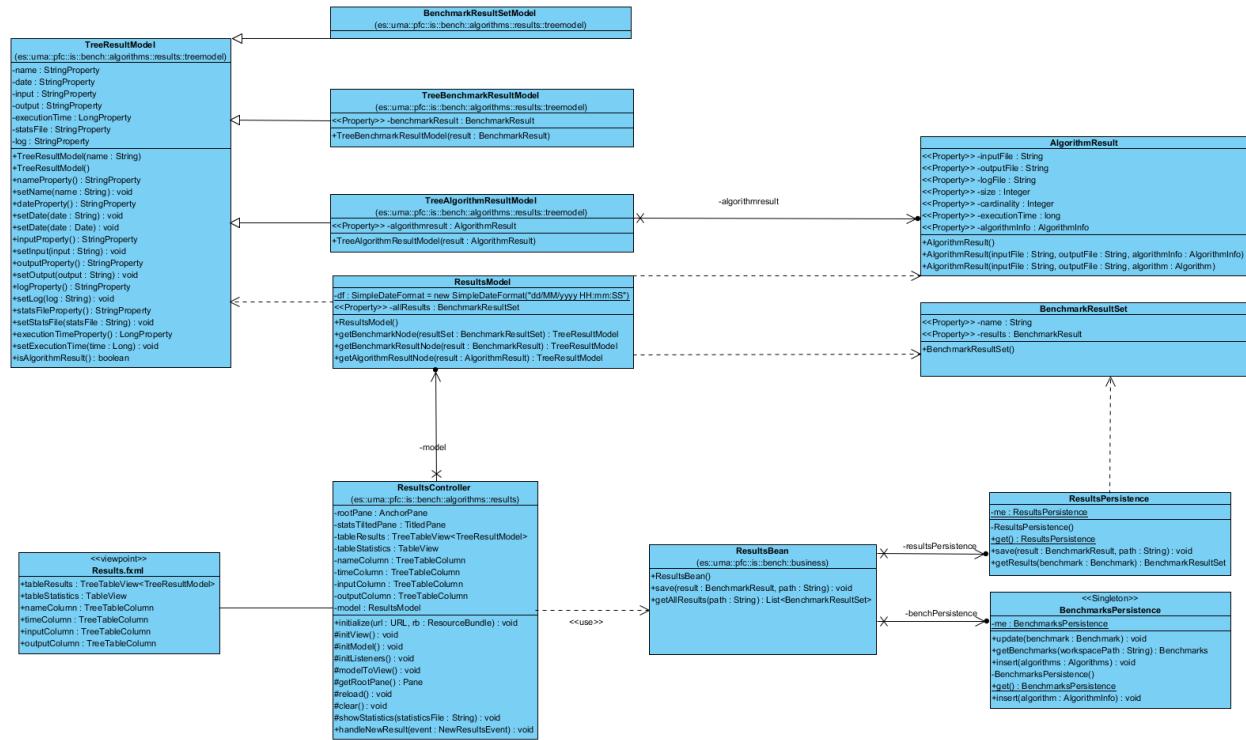
Name	Value
Description	Implements the methods to save implicational systems in files.
Visibility	public

Operations

<u>public save (systems : java.util.List< fr.kbertet.lattice.ImplicationalSystem>, fileName : String) : void</u>		
Parameters	systems Description Implicational systems. Multiplicity 1 Type java.util.List < fr.kbertet.lattice.Implicational System> Direction inout	
	fileName Description File names prefix. Multiplicity Unspecified Type ● String Direction inout	
Description	Java Detail	
Exceptions	java.io.IOException if read / write error occur.	

<u>protected checkFile (fileName : String) : void</u>		
Parameters	fileName Description File name. Multiplicity 1 Type ● String Direction inout	
Description	Checks if the file with name fileName parameter value exists. If no exists, this will be created.	
Exceptions	java.io.IOException if read / write error occur.	

6.5.7. Resultados



Summary

Name	Description
Results.fxml	Results view.
ResultsController	Results view controller.
ResultsModel	Results view model.
TreeResultModel	Results tree table view model. Represents a row of the results table.
TreeBenchmarkResultModel	Model for benchmark result node.
TreeAlgorithmResultModel	Model for algorithm result nodes.
BenchmarkResultSetModel	Benchmark result set tree node model.
AlgorithmResult	Algorithm result info.
BenchmarkResultSet	Collection of benchmark results.
ResultsBean	Results bean.
ResultsPersistence	Class for the read and write results to/from files.
BenchmarksPersistence	Persists the benchmarks into an XML file entities using JAXB. The details can be found in the <i>Registrar Benchmarks</i> diagram class.

Details



Results.fxml

Name	Value
Visibility	public
Stereotypes	viewpoint

Attributes

public tableResults : javafx.scene.control.TreeTableView			
Description	Table which contains the results registered in the current workspace, with a tree hierarchy.		
Type	javafx.scene.control.TreeTableView		
Getter	false	Setter	false
Multiplicity	1		

public tableStatistics : TableView			
Description	Table which contains the statistics of the results registered in the current workspace.		
Type	javafx.scene.control.TableView		
Getter	false	Setter	false
Multiplicity	1		



ResultsController

Name	Value
Description	Results view controller.
Visibility	public

Attributes

private rootPane : javafx.scene.layout.AnchorPane	
Description	Root pane.

Type	javafx.scene.layout.AnchorPane		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private statsTitledPane : javafx.scene.control.TitledPane			
Description	Pane which contains the statistics table.		
Type	javafx.scene.control.TitledPane		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private tableResults : javafx.scene.control.TreeTableView			
Description	Table which contains the benchmarks execution results in a tree hierarchy.		
Type	javafx.scene.control.TreeTableView		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private tableStatistics : javafx.scene.control.TableView			
Type	javafx.scene.control.TableView		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private nameColumn : javafx.scene.control.TreeTableColumn			
Type	javafx.scene.control.TreeTableColumn		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private timeColumn : javafx.scene.control.TreeTableColumn			
Type	javafx.scene.control.TreeTableColumn		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private inputColumn : javafx.scene.control.TreeTableColumn			
Type	javafx.scene.control.TreeTableColumn		
Getter	false	Setter	false
Derived	false		
Multiplicity	1		
Stereotypes	FXML		

private outputColumn : javafx.scene.control.TreeTableColumn			
Type	javafx.scene.control.TreeTableColumn		
Getter	false	Setter	false
Multiplicity	1		
Stereotypes	FXML		

private model : ResultsModel			
Description	Results view model.		
Type	 ResultsModel		
Getter	false	Setter	false
Multiplicity	1		

Operations

protected getRootPane () : javafx.scene.layout.Pane	
Description	Root pane.
Return Type Description	Pane.

public initialize (url : java.net.URL, rb : java.util.ResourceBundle) : void

	url
Description	URL of the view.
Multiplicity	1
Type	java.net.URL
Direction	inout
	rb
Description	Resource bundle.
Multiplicity	1
Type	java.util.ResourceBundle
Direction	inout
Description	Initializes the view.

protected initView () : void

Description	Initializes the results table.
Upper	1
Exceptions	IOException

protected initModel () : void

Description	Initializes the model.
-------------	------------------------

protected initListeners () : void

Description	Initializes the listeners.
-------------	----------------------------

protected modelToView () : void

Description	Loads the model data into the results table.
-------------	--

protected reload () : void

Description	Reloads the view and the model.
-------------	---------------------------------

protected clear () : void

Description	Clear the tables.
-------------	-------------------

protected showStatistics (statisticsFile : String) : void	
Parameters	statisticsFile
	Description Statistics file path.
	Multiplicity 1
	Type  String
	Direction inout
Description	Shows the result statistics into a table.

public handleNewResult (event : NewResultsEvent) : void	
Parameters	event
	Description Event.
	Multiplicity 1
	Type  NewResultsEvent
	Direction inout
Description	Handles the NewResultsEvent published by Eventbus and reload the view and the model.

ResultsModel

Name	Value
Description	Results view model.
Visibility	public

Attributes

<u>private df : java.text.SimpleDateFormat</u>			
Description	Date formatter for date column values.		
Initial Value	new SimpleDateFormat("dd/MM/yyyy HH:mm:ss")		
Type	java.text.SimpleDateFormat		
Getter	false	Setter	false
Multiplicity	1		

private allResults : BenchmarkResultSet			
Description	All results.		
Stereotypes	Property		
Type	 BenchmarkResultSet		
Getter	true	Setter	true
Multiplicity	0..*		

Operations

public ResultsModel ()	
Description	Constructor.

public getBenchmarkNode (resultSet : BenchmarkResultSet) : TreeResultModel	
Parameters	resultSet
	Description
	Benchmark result set.
	Multiplicity
	1
	Type
	 BenchmarkResultSet
	Direction
	inout
Description	Returns a Benchmark node with a result set values passed by parameter.
Return Type Description	Benchmark node.

public getBenchmarkResultNode (result : es.uma.pfc.is.bench.domain.BenchmarkResult) : TreeResultModel	
Parameters	result
	Description
	Benchmark result.
	Multiplicity
	1
	Type
	es.uma.pfc.is.bench.domain.BenchmarkResult
	Direction
	inout
Description	Creates a tree node with a benchmark result values.
Return Description Type	A tree node with a benchmark result values.

public getAlgorithmResultNode (result : AlgorithmResult) : TreeResultModel	
Parameters	result
	Description Algorithm execution result.
	Multiplicity 1
	Type  AlgorithmResult
	Direction inout
Description	Creates a tree node with an algorithm execution result values.
Return Type Description	A tree node with a algorithm result values.

TreeResultModel

Name	Value
Description	Results tree table view model. Represents a row of the results table.
Visibility	public

Attributes

private name : javafx.beans.property.StringProperty			
Description	Name column binding property.		
Type	javafx.beans.property.StringProperty		
Getter	false	Setter	false
Multiplicity	1		

private date : javafx.beans.property.StringProperty			
Description	Date column binding property.		
Type	javafx.beans.property.StringProperty		
Getter	false	Setter	false
Multiplicity	1		

private input : javafx.beans.property.StringProperty

Description	Input column binding property.		
Type	javafx.beans.property.StringProperty		
Getter	false	Setter	false
Multiplicity	1		

private output : javafx.beans.property.StringProperty

Description	Output column binding property.		
Type	javafx.beans.property.StringProperty		
Getter	false	Setter	false
Multiplicity	1		

private executionTime : javafx.beans.property.LongProperty

Description	Execution time column binding property.		
Type	javafx.beans.property.LongProperty		
Getter	false	Setter	false
Multiplicity	1		

private statsFile : javafx.beans.property.StringProperty

Description	Statistics file binding property.		
Type	javafx.beans.property.StringProperty		
Getter	false	Setter	false
Multiplicity	1		

private log : javafx.beans.property.StringProperty

Description	Log file binding property.		
Type	javafx.beans.property.StringProperty		
Getter	false	Setter	false
Multiplicity	1		

Operations

public TreeResultModel (name : String)	
Parameters	name
	Description Name.
	Multiplicity 1
	Type  String
	Direction inout
Description	Constructor.

public TreeResultModel ()	
Description	Constructor.

public nameProperty () : javafx.beans.property.StringProperty	
Description	Returns the name column binding property.
Return Type Description	The name column binding property.

public setName (name : String) : void	
Parameters	name
	Description Name.
	Multiplicity 1
	Type  String
	Direction inout
Description	Establishes the name column value.

public dateProperty () : javafx.beans.property.StringProperty	
Description	Returns the date column binding property.
Return Type Description	The date column binding property.

public inputProperty () : javafx.beans.property.StringProperty	
Description	Returns the input column binding property.
Return Type Description	The input column binding property..

public setInput (input : String) : void

Parameters	output	
	Description	Input.
	Multiplicity	1
	Type	String
	Direction	inout
Description	Establishes the input column value.	

public setDate (date : String) : void

Parameters	date	
	Description	Date string.
	Multiplicity	1
	Type	String
	Direction	inout
Description	Establishes the date column value from a string value.	

public setDate (date : java.util.Date) : void

Parameters	date	
	Description	Date.
	Multiplicity	1
	Type	java.util.Date
	Direction	inout
Description	Establishes the date with the format "dd/MM/yyyy HH:mm:SS".	

public outputProperty () : javafx.beans.property.StringProperty

Description	Returns the output column binding property.
Return Type Description	The output column binding property..

public logProperty () : javafx.beans.property.StringProperty

Description	Returns the log binding property.
Return Type Description	The log binding property.

public setOutput (output : String) : void	
Parameters	output
	Description Output.
	Multiplicity 1
	Type  String
	Direction inout
Description	Establishes the output column value

public setLog (log : String) : void	
Parameters	log
	Description Log path.
	Multiplicity 1
	Type  String
	Direction inout
Description	Establishes the execution log path.

public statsFileProperty () : javafx.beans.property.StringProperty	
Description	Returns the statistics file binding property.
Return Type Description	The statistics file binding property..

public isAlgorithmResult () : boolean	
Description	If the row is an algorithm result.
Return Type Description	<code>true</code> if is an algorithm result, <code>false</code> otherwise.

public setStatsFile (statsFile : String) : void	
Parameters	statsFile
	Description Statistics file path.
	Multiplicity 1
	Type  String
	Direction inout
Description	Establishes the statistics file path.

public executionTimeProperty () : javafx.beans.property.LongProperty

Description	Returns the execution time column binding property.
Return Type Description	The execution time column binding property..

public setExecutionTime (time : Long) : void

Parameters	time	
	Description	Execution time.
	Multiplicity	1
	Type	Long
	Direction	inout
Description	Establishes the execution time.	

TreeBenchmarkResultModel

Name	Value
Description	Model for benchmark result node.
Visibility	public

Attributes

private benchmarkResult : BenchmarkResult

Description	Benchmark result.		
Stereotypes	Property		
Type	 BenchmarkResult		
Getter	true	Setter	false
Multiplicity	1		

Operations

public TreeBenchmarkResultModel (result : es.uma.pfc.is.bench.domain.BenchmarkResult)	
Parameters	result
	Description Benchmark result.
	Multiplicity 1
	Type es.uma.pfc.is.bench.domain.BenchmarkResult
	Direction inout
Description	Constructor. Initializes the node with a benchmark result values.

TreeAlgorithmResultModel

Name	Value
Description	Model for algorithm result nodes.
Visibility	public

Attributes

private algorithmresult : AlgorithmResult			
Description	Algorithm result.		
Stereotypes	Property		
Type	 AlgorithmResult		
Getter	true	Setter	false
Derived	false		
Multiplicity	1		

Operations

public TreeAlgorithmResultModel (result : AlgorithmResult)	
Parameters	result
	Description Algorihtm result.
	Multiplicity 1
	Type  AlgorithmResult
	Direction inout
Description	Constructor.

BenchmarkResultSetModel

Name	Value
Description	Benchmark result set tree node model.
Visibility	public

BenchmarkResultSet

Name	Value
Description	Collection of benchmark results.
Visibility	public

Operations

public BenchmarkResultSet ()	
Description	Constructor.

Attributes

private name : String			
Description	Benchmark name.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		

private results : BenchmarkResult			
Description	Benchmark results.		
Stereotypes	Property		
Type	 BenchmarkResult		
Getter	true	Setter	true
Multiplicity	0..*		



AlgorithmResult

Name	Value
Description	Algorithm result info.
Visibility	public

Attributes

private inputFile : String			
Description	Implicational system input file.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	false
Multiplicity	1..*		

private outputFile : String			
Description	Implicational system output file.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	false
Multiplicity	1		

private logFile : String			
Description	Log file.		
Stereotypes	Property		
Type	 String		
Getter	true	Setter	true
Multiplicity	1		

private size : Integer			
Description	Implicational System size.		
Stereotypes	Property		
Type	Integer		
Getter	true	Setter	false
Multiplicity	1		

private cardinality : Integer			
Description	Implicational System cardinality.		
Stereotypes	Property		
Type	Integer		
Getter	true	Setter	false
Multiplicity	1		

private executionTime : long			
Description	Execution time.		
Stereotypes	Property		
Type	long		
Getter	true	Setter	true
Multiplicity	1		

private algorithmInfo : AlgorithmInfo			
Description	Info of the executed algorithm.		
Stereotypes	Property		
Type	AlgorithmInfo		
Getter	true	Setter	false
Multiplicity	1		

Operations

public AlgorithmResult ()	
Description	Constructor.

public AlgorithmResult (inputFile : String, outputFile : String, algorithm : Algorithm)		
Parameters	inputFile	
	Description	Input system file.
	Multiplicity	1
	Type	String
	Direction	inout
	outputFile	
	Description	Output system file.
	Multiplicity	1
	Type	String
	Direction	inout
	algorithm	
	Description	Algorithm.
	Multiplicity	1

public AlgorithmResult (inputFile : String, outputFile : String, algorithm : Algorithm)		
	Type	 Algorithm
	Direction	inout
Description	Constructor.	

public AlgorithmResult (inputFile : String, outputFile : String, algorithmInfo : AlgorithmInfo)		
Parameters	inputFile	
	Description	Input system file.
	Multiplicity	1
	Type	 String
	Direction	inout
outputFile		
	Description	Output system file.
	Multiplicity	1
	Type	 String
	Direction	inout
algorithmInfo		
	Description	Algorithm info.
	Multiplicity	1
	Type	 AlgorithmInfo
	Direction	inout
Description	Constructor.	

ResultsBean

Name	Value
Description	Logic for reads and insert benchmark results in files.
Visibility	public

Attributes

private benchPersistence : BenchmarksPersistence			
Description	Benchmarks persistence.		
Type	 BenchmarksPersistence		
Getter	false	Setter	false
Multiplicity	1		

private resultsPersistence : ResultsPersistence			
Description	Results persistence.		
Type	 ResultsPersistence		
Getter	false	Setter	false
Multiplicity	1		

Operations

public ResultsBean ()	
Description	Constructor.

public save (result : es.uma.pfc.is.bench.domain.BenchmarkResult, path : String) : void	
Parameters	result
	Description
	Benchmark result.
	Multiplicity
	1
	Type
	es.uma.pfc.is.bench.domain.BenchmarkResult
	Direction
	inout
	path
	Description
	Directory of results.xml.
	Multiplicity
	1
	Type
	 String
	Direction
	inout
Description	Saves a benchmark result in results.xml file.

public getAllResults (path : String) : java.util.List	
Parameters	path
	Description Path of registered benchmarks.
	Multiplicity 1
	Type  String
	Direction inout
Description	Returns all saved results of the registered benchmarks in the path.
Return Type Description	Results saved.

ResultsPersistence

Name	Value
Description	Class for the read and write results to/from files.
Visibility	public

Attributes

<u>private me : ResultsPersistence</u>			
Description	Single instance.		
Type	 ResultsPersistence		
Getter	false	Setter	false
Multiplicity	1		

Operations

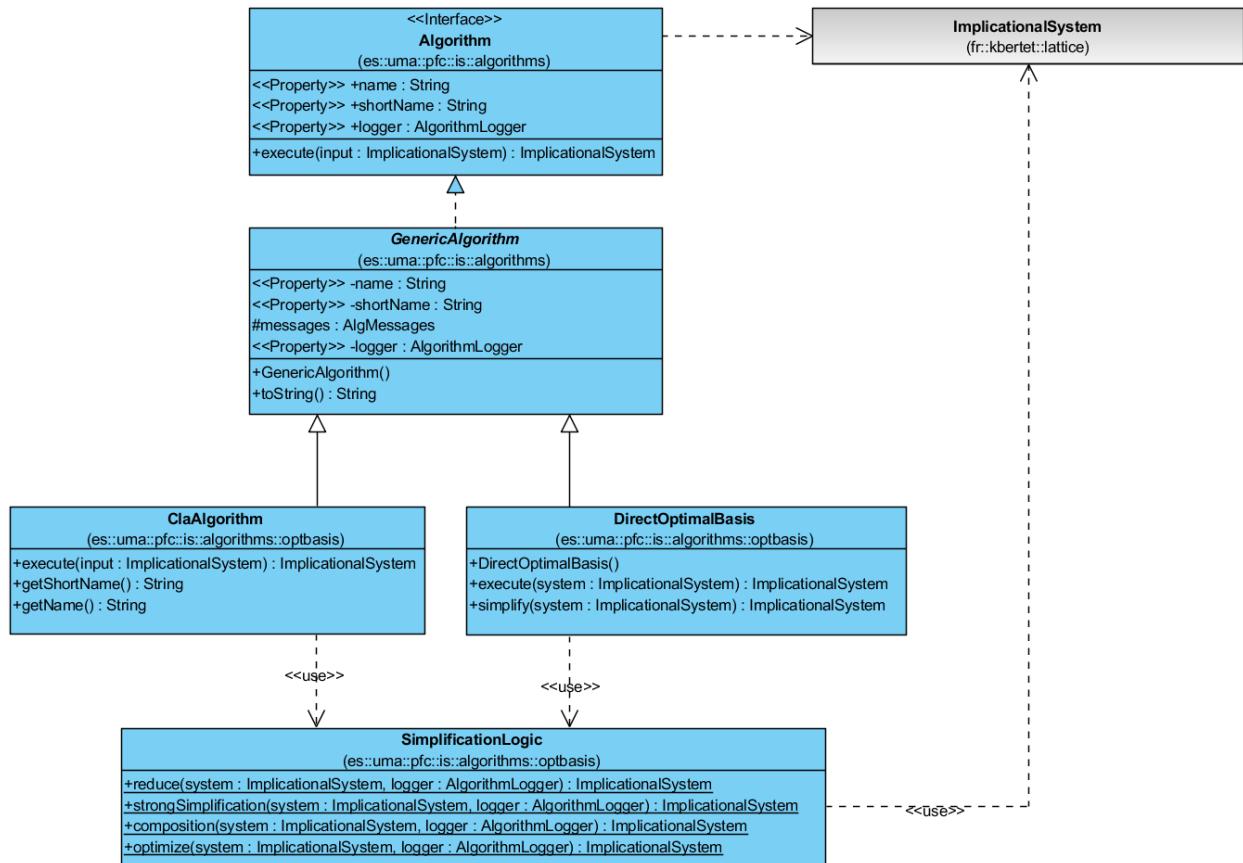
<u>private ResultsPersistence ()</u>	
Description	Constructor.

<u>public get () : ResultsPersistence</u>	
Description	Gets a single instance of ResultsPersistence.
Return Type Description	ResultsPersistence single instance.

public getResults (benchmark : Benchmark) : BenchmarkResultSet	
Parameters	benchmark
	Description
	Multiplicity
	Type
	Direction
Description	Returns a benchmark's saved results.
Return Type Description	Benchmark's saved results. null if the results not exists.

public save (result : es.uma.pfc.is.bench.domain.BenchmarkResult, path : String) : void	
Parameters	result
	Description
	Multiplicity
	Type
	Direction
Parameters	path
	Description
	Multiplicity
	Type
	Direction
Description	Saves a benchmark result to the results.xml file in a directory passed by parameter.

6.5.8. API Algoritmos



Summary

Name	Description
Algorithm	Algorithm of implicative system basis computation.
GenericAlgorithm	Generic algorithm which receive as input a file input and returns an implicative system.
DirectOptimalBasis	AMIS Algorithm implementation.
ClaAlgorithm	CLA Algorithm implementation.
ImplicationalSystem	This class gives a representation for an implicative system (ImplicationalSystem), a set of rules. This class belong to java-lattices library.
SimplificationLogic	Methods which implements the simplification logic rules.

Description

API for implementation of algorithms which can be executed by IS Bench.

Details



Algorithm

Name	Value
Description	Algorithm of implicational system basis computation. This interface must be implemented by the classes which will be executed by IS Bench.
Visibility	public

Attributes

public name : String			
Description	Algorithm name. It will be used as the algorithm string representation.		
Stereotypes	Property		
Type	String		
Getter	true	Setter	true
Multiplicity	1		

public shortName : String			
Description	Algorithm short name. The short name will be used as the default name of the output files.		
Stereotypes	Property		
Type	String		
Getter	true	Setter	true
Multiplicity	1		

public logger : es.uma.pfc.is.logging.AlgorithmLogger			
Description	Logger for trace generation.		
Stereotypes	Property		
Type	es.uma.pfc.is.logging.AlgorithmLogger		
Getter	true	Setter	false
Multiplicity	1		

Operations

public execute (input : fr.kbertet.lattice.ImplicationalSystem) : fr.kbertet.lattice.ImplicationalSystem .ImplicationalSystem	
Parameters	input Description Input implicational system. Multiplicity 1 Type fr.kbertet.lattice.ImplicationalSystem Direction inout
Description	Executes the algorithm with the implicational system parameter as input.
Return Type Description	Implicational System returned by the executed algorithm.

GenericAlgorithm

Name	Value
Description	Generic algorithm which receive as input a file input and returns an implicational system.
Visibility	public

Attributes

private name : String			
Description	Name.		
Stereotypes	Property		
Type	String		
Getter	true	Setter	true
Multiplicity	1		

private shortName : String			
Description	Short name.		
Stereotypes	Property		
Type	String		
Getter	true	Setter	true
Multiplicity	1		

protected messages : AlgMessages

Description	Translates messages to current language.		
Type	 AlgMessages		
Getter	false	Setter	false
Multiplicity	1		

private logger : es.uma.pfc.is.logging.AlgorithmLogger

Description	Logger.		
Stereotypes	Property		
Type	es.uma.pfc.is.logging.AlgorithmLogger		
Getter	true	Setter	false
Multiplicity	1		

Operations**public GenericAlgorithm ()**

Description	Constructor.
-------------	--------------

protected setLogger (logger : es.uma.pfc.is.logging.AlgorithmLogger) : void

Parameters	logger	
	Description	Logger.
	Multiplicity	1
	Type	es.uma.pfc.is.logging.AlgorithmLogger
	Direction	inout
Description	For testing usage.	

protected removeRule (system : fr.kbertet.lattice.ImplicationalSystem, rule : fr.kbertet.lattice.Rule) : void		
Parameters	system Description Implicational system. Multiplicity 1 Type fr.kbertet.lattice.ImplicationalSystem Direction inout	
	rule Description Rule. Multiplicity 1 Type fr.kbertet.lattice.Rule Direction inout	
Description	Removes rules for implicational system, and print trace in the log.	

protected addRule (system : fr.kbertet.lattice.ImplicationalSystem, rule : fr.kbertet.lattice.Rule) : void		
Parameters	system Description Implicational system. Multiplicity 1 Type fr.kbertet.lattice.ImplicationalSystem Direction inout	
	rule Description Rule. Multiplicity 1 Type fr.kbertet.lattice.Rule Direction inout	
Description	Adds rules for implicational system, and print trace in the log.	
Return Type Description	Implicational system with a rule added.	

protected addRuleAndElements (system : fr.kbertet.lattice.ImplicationalSystem, rule : fr.kbertet.lattice.Rule) : ImplicationalSystem

Parameters	system	
	Description	Implicational system.
	Multiplicity	1
	Type	fr.kbertet.lattice.ImplicationalSystem
	Direction	inout
	rule	
	Description	Rule.
	Multiplicity	1
	Type	fr.kbertet.lattice.Rule
	Direction	inout
Description	Adds rules for implicational system, and print trace in the log.	
Return Type Description	Implicational system with a rule and its elements added.	

protected addRuleAndElements (system : fr.kbertet.lattice.ImplicationalSystem, rule : fr.kbertet.lattice.Rule, trace : boolean) : fr.kbertet.lattice.ImplicationalSystem

Parameters	system	
	Description	Implicational system.
	Multiplicity	1
	Type	fr.kbertet.lattice.ImplicationalSystem
	Direction	inout
	rule	
	Description	Rule.
	Multiplicity	1
	Type	fr.kbertet.lattice.Rule
	Direction	inout
trace		
Description	If print trace in the log.	
Multiplicity	1	
Type	boolean	
Direction	inout	
Description	Adds rules for implicational system and its elements, and print trace in the log if the trace parameter is true.	

protected history (message : String, args : Object) : void

Parameters	message
	Description Message.
	Multiplicity 1
	Type String
	Direction inout
Parameters	args
	Description Message arguments.
	Multiplicity 0..*
	Type Modifier ...
	Type Object
	Direction inout
Description	Prints a message with the arguments, to the log.

**protected replaceRule (system : fr.kbertet.lattice.ImplicationalSystem,
rule1 : fr.kbertet.lattice.Rule,
rule2 : fr.kbertet.lattice.Rule) : void**

Parameters	system
	Description Implicational system.
	Multiplicity 1
	Type fr.kbertet.lattice.ImplicationalSystem
	Direction inout
Parameters	rule1
	Description Rule to replace.
	Multiplicity 1
	Type fr.kbertet.lattice.Rule
	Direction inout
Parameters	rule2
	Description New rule.
	Multiplicity 1
	Type fr.kbertet.lattice.Rule
	Direction inout

Description	Replace a rule by other for implicational system, and print trace in the history.
-------------	---

public <code>toString () : String</code>	
Description	Algorithm string representation. By default, is the name property value.
Return Type Description	Name.

DirectOptimalBasis

Name	Value
Description	Direct Optimal Basis algorithm implementation.
Visibility	public

Operations

public <code>DirectOptimalBasis ()</code>	
Description	Constructor.

public <code>execute (system : ImplicationalSystem) : ImplicationalSystem</code>	
Parameters	system
	Description Input system.
	Multiplicity 1
	Type  ImplicationalSystem
	Direction inout
Description	Executes the Direct Optimal Basis algorithm.

public simplify (system : ImplicationalSystem) : ImplicationalSystem		
Parameters	system	
	Description	Reduced system.
	Multiplicity	1
	Type	 ImplicationalSystem
	Direction	inout
Description	Generation of IS simplified by simplification(left+right+composition) of reduced IS	
Return Type Description	Simplified system.	
Query	false	

protected printInit (inputSystem : fr.kbertet.lattice.ImplicationalSystem) : void		
Parameters	inputSystem	
	Description	Implicactional System.
	Multiplicity	Unspecified
	Type	fr.kbertet.lattice.ImplicationalSystem
	Direction	inout
Description	Prints de initial arguments.	

protected printResult (resultSystem : fr.kbertet.lattice.ImplicationalSystem) : void		
Parameters	resultSystem	
	Description	Implicational System.
	Multiplicity	Unspecified
	Type	fr.kbertet.lattice.ImplicationalSystem
	Direction	inout
Description	Prints the results.	

ClaAlgorithm

Name	Value
Description	CLA Algorithm implementation.
Visibility	public

Operations

public execute (input : ImplicationalSystem) : ImplicationalSystem	
Parameters	input Description Multiplicity Type Direction
Description	Executes the CLA algorithm for computation of the input implicational system direct basis. Uses the SimplificationLogic class methods.

public getShortName () : String	
Description	Short name.

public getName () : String	
Description	Name.



SimplificationLogic

Name	Value
Description	Methods which implements the simplification logic rules.
Visibility	public

Operations

public fragmentationEquivalency (implication : fr.kbertet.lattice.Rule) : fr.kbertet.lattice.Rule	
Parameters	implication Description Multiplicity Type Direction
Description	Implements the Fragmentation Equivalency rule: $[FrEq]: \{A \rightarrow B\} = \{A \rightarrow B - A\}$.
Return Type Description	Equivalent simplified implication.

<u>public compositionEquivalency (rule1 : fr.kbertet.lattice.Rule, rule2 : fr.kbertet.lattice.Rule) :java.util.List< fr.kbertet.lattice.Rule></u>																	
Parameters	rule1 <table border="1"> <tr> <td>Description</td><td>Implication.</td></tr> <tr> <td>Multiplicity</td><td>1</td></tr> <tr> <td>Type</td><td>fr.kbertet.lattice.Rule</td></tr> <tr> <td>Direction</td><td>inout</td></tr> </table> rule2 <table border="1"> <tr> <td>Description</td><td>Implication.</td></tr> <tr> <td>Multiplicity</td><td>1</td></tr> <tr> <td>Type</td><td>fr.kbertet.lattice.Rule</td></tr> <tr> <td>Direction</td><td>inout</td></tr> </table>	Description	Implication.	Multiplicity	1	Type	fr.kbertet.lattice.Rule	Direction	inout	Description	Implication.	Multiplicity	1	Type	fr.kbertet.lattice.Rule	Direction	inout
Description	Implication.																
Multiplicity	1																
Type	fr.kbertet.lattice.Rule																
Direction	inout																
Description	Implication.																
Multiplicity	1																
Type	fr.kbertet.lattice.Rule																
Direction	inout																
Description	Given two implications: $\{A \rightarrow B, A \rightarrow C\} \Rightarrow \{A \rightarrow BC\}$																
Return Type Description	If the composition rule can be applied, returns a list with once implication from the rule application. If the composition rule can't be applied, returns a list which contains the rules passed as parameters.																
Exceptions	java.lang.NullPointerException Si alguna de las implicaciones es nula.																

<u>public rightSimplificationEq (rule1 : fr.kbertet.lattice.Rule, rule2 : fr.kbertet.lattice. Rule) : java.util.List< fr.kbertet.lattice.></u>																	
Parameters	rule1 <table border="1"> <tr> <td>Description</td><td>Implication.</td></tr> <tr> <td>Multiplicity</td><td>1</td></tr> <tr> <td>Type</td><td>fr.kbertet.lattice.Rule</td></tr> <tr> <td>Direction</td><td>inout</td></tr> </table> rule2 <table border="1"> <tr> <td>Description</td><td>Implication.</td></tr> <tr> <td>Multiplicity</td><td>1</td></tr> <tr> <td>Type</td><td>fr.kbertet.lattice.Rule</td></tr> <tr> <td>Direction</td><td>inout</td></tr> </table>	Description	Implication.	Multiplicity	1	Type	fr.kbertet.lattice.Rule	Direction	inout	Description	Implication.	Multiplicity	1	Type	fr.kbertet.lattice.Rule	Direction	inout
Description	Implication.																
Multiplicity	1																
Type	fr.kbertet.lattice.Rule																
Direction	inout																
Description	Implication.																
Multiplicity	1																
Type	fr.kbertet.lattice.Rule																
Direction	inout																
Description	Implements the simplification rule for two implications.																

public rightSimplificationEq (rule1 : fr.kbertet.lattice.Rule, rule2 : fr.kbertet.lattice. Rule) : java.util.List< fr.kbertet.lattice.>
--

	[SiEq]: if (A intersection B is empty) and (A subset of C) then {A -> B, C -> D} == {A -> B, C-B -> D-B}
Return Type Description	Simplified implications.
Exceptions	java.lang.NullPointerException if any rule is null.

public compositionEquivalency (is : fr.kbertet.lattice.ImplicationalSystem, rule1 : fr.kbertet.lattice.Rule, rule2 : fr.kbertet.lattice.Rule) : fr.kbertet.lattice.ImplicationalSystem

Parameters	is
	Description Implicational system.
	Multiplicity 1
	Type fr.kbertet.lattice.ImplicationalSystem
	Direction inout
	rule1
	Description Implication.
	Multiplicity 1
	Type fr.kbertet.lattice.Rule
	Direction inout
	rule2
	Description Implication.
	Multiplicity 1
	Type fr.kbertet.lattice.Rule
	Direction inout
Description	rule1 and rule2 are implications from is implicational system parameter. If the composition rule can be applied to rule1 and rule2, removes these rules from the implicational system and adds the new.

```
public strongSimplificationEq (rule1 : fr.kbertet.lattice.Rule,
                               rule2 : fr.kbertet.lattice.Rule)
                               : fr.kbertet.lattice.Rule
```

Parameters	rule1	
	Description	Implication.
	Multiplicity	1
	Type	fr.kbertet.lattice.Rule
	Direction	inout
	rule2	
	Description	Implicación.
	Multiplicity	1
	Type	fr.kbertet.lattice.Rule
	Direction	inout
Description	If (B intersection C) not is empty and (D \ A union B)) neither, returns the new implication AC - B -> D - (AB).	
Return Type Description	Returns a new implication if Strong Simplification can be applied, null otherwise.	

```
public reduce (system : ImplicationalSystem, logger : AlgorithmLogger)
               : fr.kbertet.lattice.ImplicationalSystem
```

Parameters	system	
	Description	Implicational system.
	Multiplicity	1
	Type	fr.kbertet.lattice.ImplicationalSystem
	Direction	inout
	logger	
	Description	Logger.
	Multiplicity	1
	Type	AlgorithmLogger
	Direction	inout
Description	Gets a reduced system. An implicational system is reduced, if A -> B => (B not empty) AND (A intersection B is empty) for all A, B in S.	
Return Type Description	Reduced implicational system.	

```
public strongSimplification (system : fr.kbertet.lattice.ImplicationalSystem,
                            logger : AlgorithmLogger)
                           : fr.kbertet.lattice.ImplicationalSystem
```

Parameters	system
	Description Simplified system.
	Multiplicity 1
	Type fr.kbertet.lattice.ImplicationalSystem
	Direction inout
	logger
	Description Logger.
	Multiplicity 1
	Type  AlgorithmLogger
	Direction inout
Description	Generation of IS by completion of simplified IS --> Strong Simplification.
Return Type Description	Strong simplified system.
Exceptions	java.lang.NullPointerException if system is null.

```
public composition (system : fr.kbertet.lattice.ImplicationalSystem,
                     logger : AlgorithmLogger)
                     : fr.kbertet.lattice. ImplicationalSystem
```

Parameters	system
	Description Implicational System.
	Multiplicity 1
	Type fr.kbertet.lattice.ImplicationalSystem
	Direction inout
	logger
	Description Loger
	Multiplicity 1
	Type  AlgorithmLogger
	Direction inout
Description	Composition of implications of a system.
Return Type Description	Implicational System with composition applied.

<u>public optimize (system : fr.kbertet.lattice.ImplicationalSystem,</u>
<u>logger : AlgorithmLogger) : fr.kbertet.lattice.ImplicationalSystem</u>

Parameters	system
	Description Simplified Implicational System.
	Multiplicity 1
	Type fr.kbertet.lattice.ImplicationalSystem
	Direction inout
	logger
	Description Logger.
	Multiplicity 1
	Type  AlgorithmLogger
	Direction inout
Description	Generation of optimized IS.
Return Type Description	Optimized system.

6.6. Diagramas de secuencia

Los diagramas de secuencia muestran el intercambio de mensajes (es decir la forma en que se invocan) en un momento dado. Ponen especial énfasis en el orden y el momento en que se envían los mensajes a los objetos.

En los diagramas de secuencia, los objetos están representados por líneas intermitentes verticales, con el nombre del objeto en la parte más alta. El eje de tiempo también es vertical, incrementándose hacia abajo, de forma que los mensajes son enviados de un objeto a otro en forma de flechas con los nombres de la operación y los parámetros.

En esta sección, se presentan los diagramas de secuencia de los escenarios más relevantes.

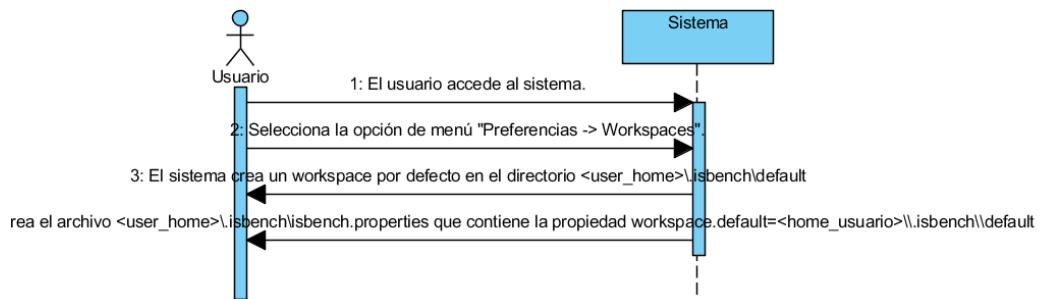


Figura 6.3: Primer acceso al sistema

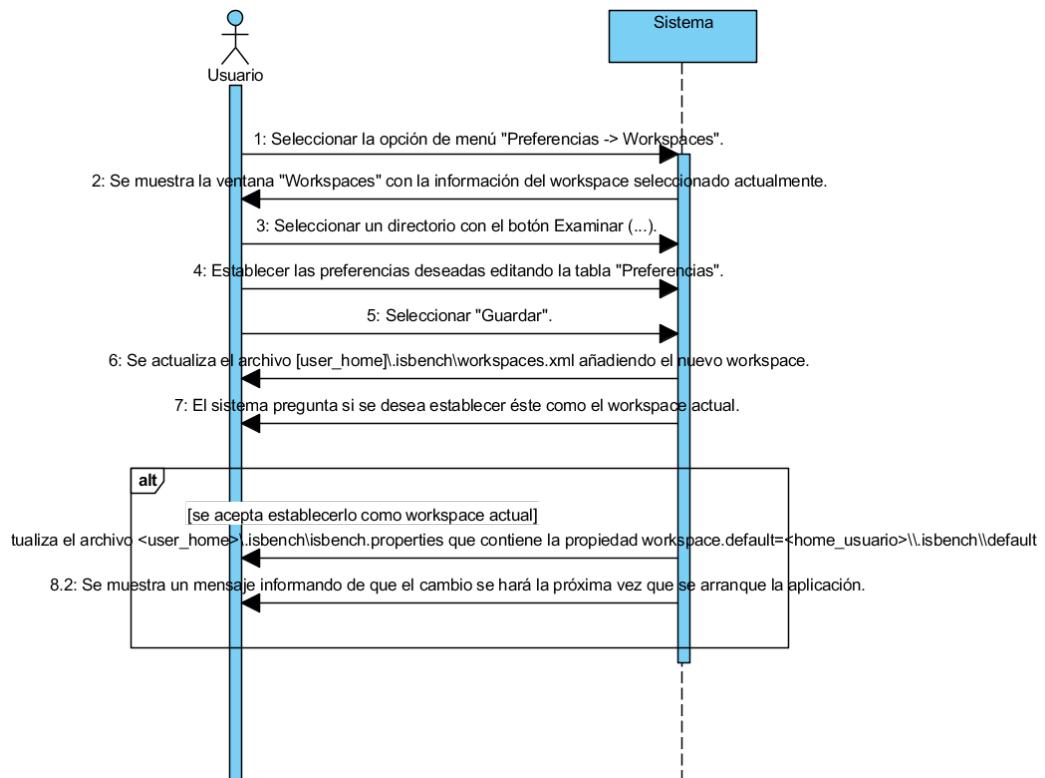


Figura 6.4: Crear un workspace

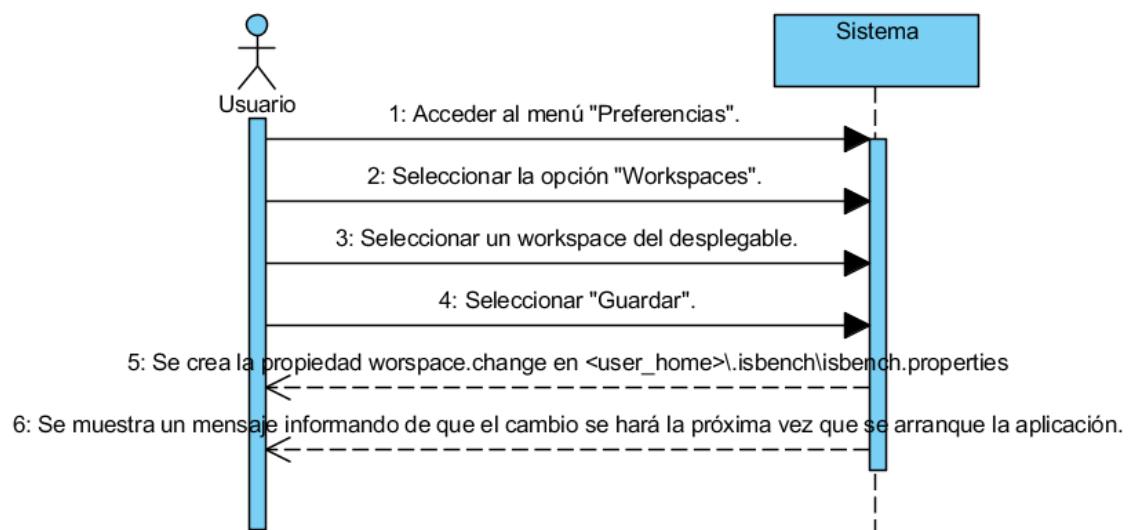


Figura 6.5: Cambiar de workspace

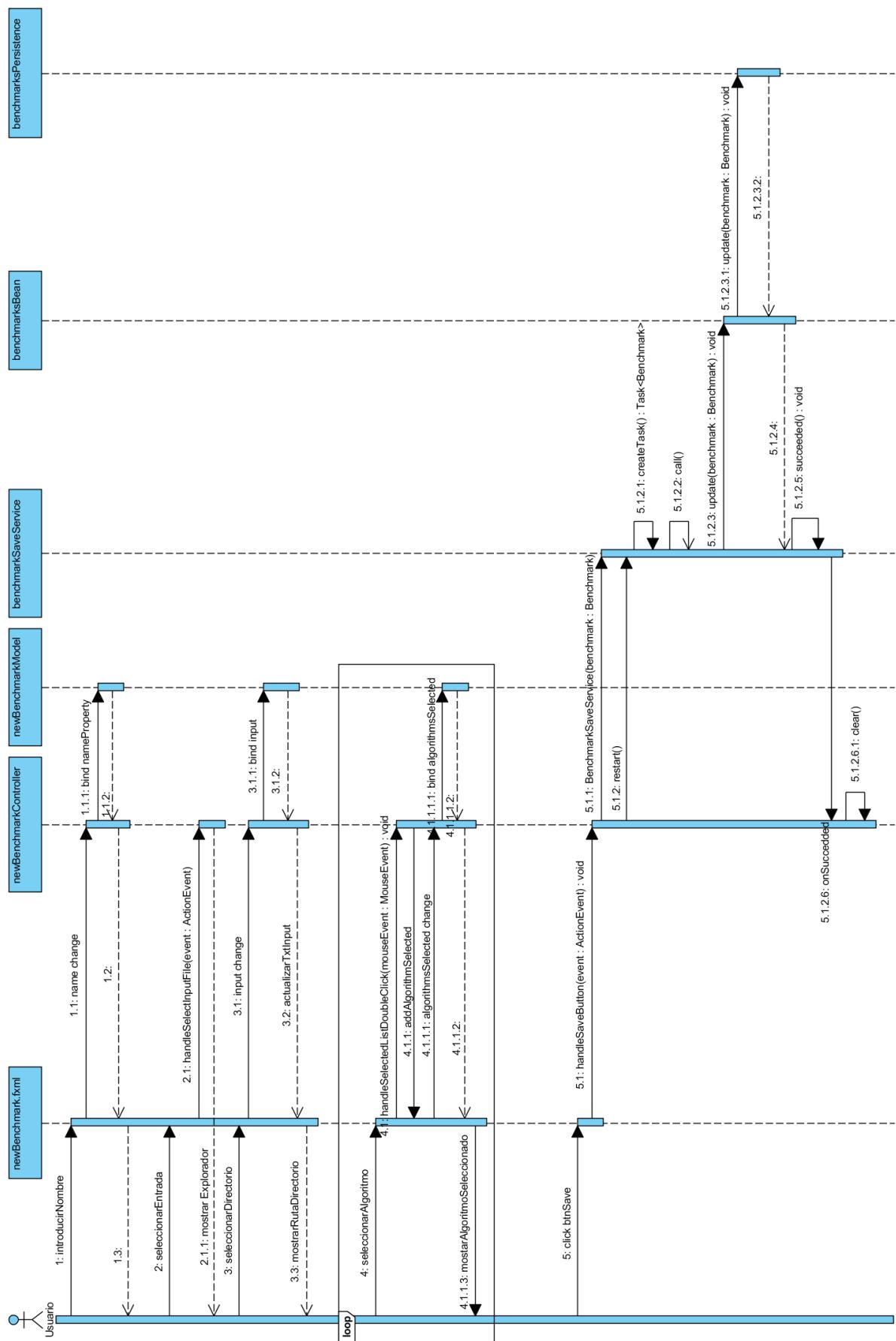


Figura 6.6: Registrar Benchmark

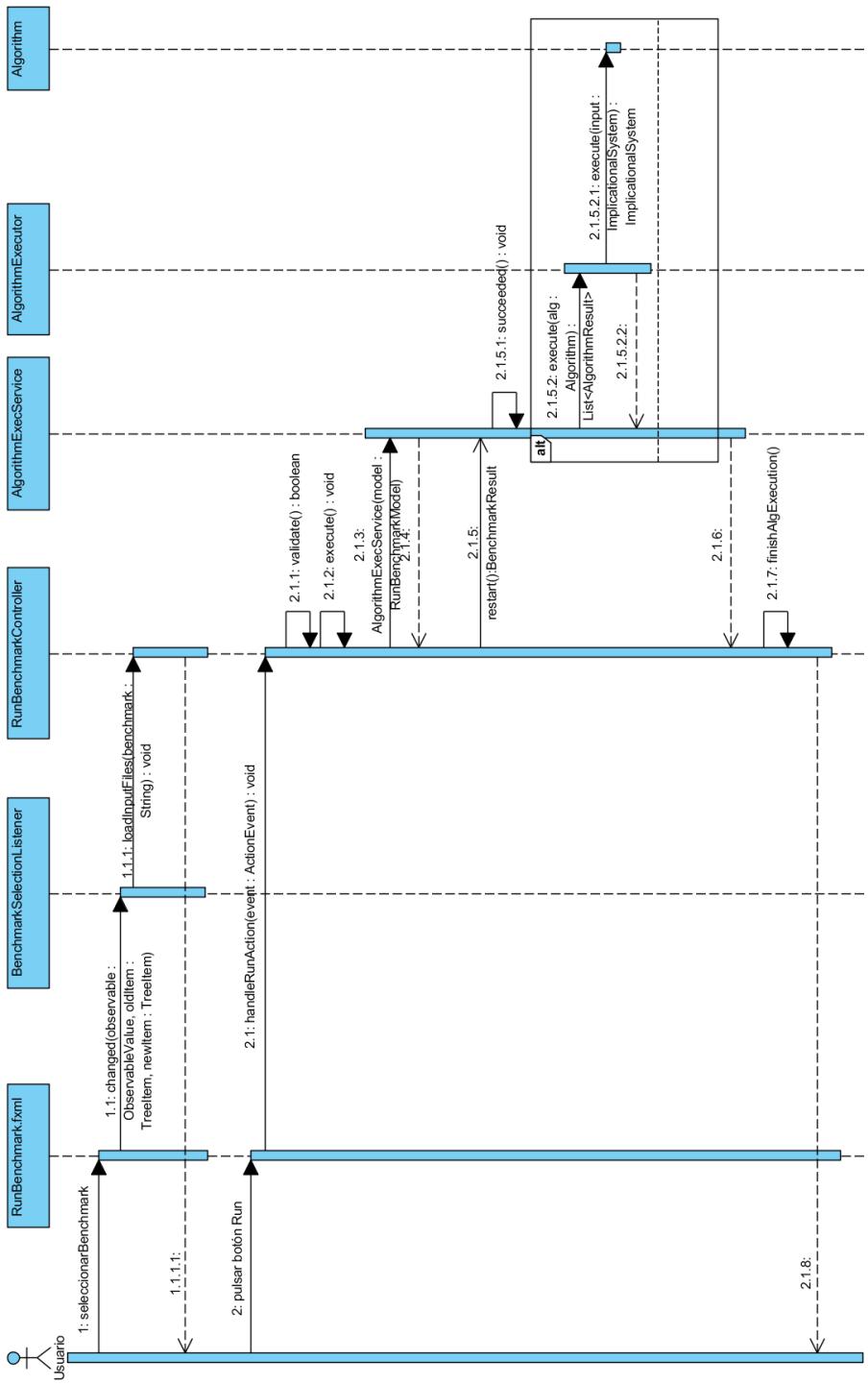


Figura 6.7: Ejecutar Benchmark

Capítulo 7

CONCLUSIONES

En este proyecto se han implementado dos herramientas de utilidad para el estudio de algoritmos sobre conjuntos de implicaciones:

- (1) Generador de implicaciones aleatorios.
- (2) IS Bench, herramienta para la ejecución de algoritmos y la posterior consulta de resultados.

Además, se han implementado dos algoritmos de cálculo de bases en Java: Direct Optimal Basis y CLA cuyos pseudocódigos se presentan en los artículos [9] y [1] respectivamente.

También, queda definida la API para la implementación de nuevos algoritmos que pueden ser ejecutados desde IS Bench.

Bibliografía

- [1] E. Rodríguez, K. Bertet, P. Cordero, M. Enciso, A. Mora y M. Ojeda-Aciego: *A method to build the Direct-Optimal Basis from an implicational system.*
- [2] K. V. Adaricheva and J. B. Nation and R. Rand, Ordered direct implicational basis of a finite closure system, International Symposium on Artificial Intelligence and Mathematics, ISAIM 2012.
- [3] K. Bertet, M. Nebut, Efficient algorithms on the Moore family associated to an implicational system, DMTCS, 6(2): 315-338, 2004.
- [4] K. Bertet, B. Monjardet, The multiple facets of the canonical direct unit implicational basis, Theor. Comput. Sci., 411(22-24): 2155-2166, 2010.
- [5] K. Bertet, Some Algorithmical Aspects Using the Canonical Direct Implicationnal Basis, CLA:101-114, 2006.
- [6] K. Bertet, C. Demko, J.F. Viaud, C. Guérin, java-lattices: a Java for lattices computation, <http://thegalactic.org>, 2014.
- [7] P Cordero, A. Mora, M. Enciso, I.Pérez de Guzmán, SLFD Logic: Elimination of Data Redundancy in Knowledge Representation, Lecture Notes in Computer Science, 2527: 141-150, 2002.
- [8] A. Mora, M. Enciso, P. Cordero, and I. Fortes, Closure via functional dependence simplification, I. J.of Computer Mathematics, 89(4): 510-526, 2012.
- [9] E. Rodríguez Lorenzo, K. Bertet, P. Cordero, M. Enciso, A. Mora, Direct-optimal Basis via Reductions, CLA: 145-156, 2014.
- [10] E. Rodríguez Lorenzo, K. Bertet, P. Cordero, M. Enciso, A. Mora, Ojeda-Aciego From Implicational Systems to Direct-Optimal bases: A logic-based Approach, Applied Mathematics & Information Sciences, 2L: 305-317, 2015.
- [11] A. Mora, M. Enciso, P. Cordero, and I. Pérez de Guzmán, Preprocessing Transformation for Functional Dependencies Sets Based on the Substitution Paradigm, Lecture Notes in Computer Science, 3040: 136-146, 2004.

- [12] Bertet, Karel and Demko, Christophe and Viaud, Jean-François and Guérin, Clément, java-lattices: a Java library for lattices computation, <http://thegalactic.github.io/java-lattices/>, 2014
- [13] Oracle, Java Platform, Standard Edition (Java SE) 8, <http://docs.oracle.com/javase/8/>
- [14] Oracle, JavaFX 2 Documentation, <http://docs.oracle.com/javafx/2/>
- [15] Oracle, Trail: Java Architecture for XML Binding, <https://docs.oracle.com/javase/tutorial/jaxb/>
- [16] QOS.ch, Simple Logging Facade for Java (SLF4J), <http://www.slf4j.org>
- [17] The Apache Software Foundation, Apache Commons CSV, <https://commons.apache.org/proper/commons-csv/>
- [18] Google, Guava, Wiki <https://github.com/google/guava/wiki>
- [19] The Apache Software Foundation, Apache Maven Project, <https://maven.apache.org/>
- [20] Object Management Group (OMG), Unified Modeling Language (UML) Resource Page, <http://www.uml.org/>