Centro de Investigación y de Estudios
Avanzados del Instituto Politécnico Nacional

Departamento de Ingeniera Eléctrica

Sección Computación

Herramienta para el Análisis y Verificación de Bases de Conocimientos descritas con el Lenguaje HAries

Tesis que presenta

Grettel Barceló Alonso

para obtener el Grado de

Maestro en Ciencias

en la Especialidad de

Ingeniera Eléctrica

Director de Tesis

Dr. José Oscar Olmedo Aguirre

Co - Director

Dr. Argelio Víctor de la Cruz Rivera

México, D.F. **Enero 2006**

Resumen

Los *Sistemas Basados en Conocimiento* (SBC) han sido utilizados en muchas áreas de aplicación donde sus fallas pueden ser muy costosas. Es por ello que ha surgido la necesidad de crear métodos de verificación que ayuden a garantizar, en cierto grado, la confiabilidad de estos sistemas. La verificación puede estar enfocada a una parte específica del sistema, como su base de conocimientos (BC) o su motor de inferencia, en dependencia de las propiedades que se analicen.

La BC es el componente más importante de un SBC. Debido a que ésta se construye generalmente de una manera incremental y no sistemática, se pueden introducir errores potenciales inadvertidamente. Por lo cual, uno de los puntos críticos al desarrollar sistemas confiables es la verificación y corrección de su BC y específicamente de las estructuras de representación del conocimiento que la conforman. Existen tres propiedades formales principales que tienen que ver con la verificación de una BC, éstas son: consistencia, redundancia y circularidad, aunque las formas en que se presentan depende de las características del lenguaje de representación empleado para construirla.

En el presente trabajo se plantea el diseño e implementación de la herramienta VBCH que analiza y verifica BC descritas con el lenguaje HAries [1]. El diseño se basa en la identificación y definición formal de las anomalías que pueden presentarse en las estructuras de representación, así como la creación de los métodos empleados por el sistema para detectarlas. Además se especifican algunas medidas cualitativas que posibilitan el análisis de características relacionadas con la estructura y complejidad de la BC.

Abstract

The *Knowledge Based Systems* (KBS) have been used in many areas of applications where their failures can be very expensive. That is the reason why it is necessary to develop verification methods to improve, in certain degree, the reliability of these systems. The verification can be restricted to a specific part of the system, such as the knowledge base (KB) or the inference engine, depending on the properties of the systems that are analyzed.

The KB is the most important component in a KBS. As this is often built in increasing and not systematic way, potential errors can be introduced inadvertently. That is why some of the critical aspects when developing reliable systems are the verification and correction of their KB and specifically of the representation structures that integrate it. There are three main formal properties dealing with the verification of a KB, and they are: consistency, redundancy and circularity, although the forms in that they are presented depends on the characteristics of the representation language used to build the base.

In this work it is presented the design and implementation of the VBCH tool that analyzes and verifies KB described with the HAries [1] language. The study begins with the identification and formal definition of the anomalies that can appear in the representation structures, as well as the methods used by the system to detect them. Besides some qualitative measures are specified to facilitate the analysis of characteristics related with the structure and complexity of the KB.

Agradecimientos

En primer lugar quiero agradecer y dedicar esta tesis a mis padres, pues ellos han sido el mejor ejemplo y apoyo en ésta y todas las etapas de mi vida. Lo que soy se lo debo a ustedes.

Al resto de mi familia, por la confianza que han tenido en mi y por el aliento que me han brindado por mi superación.

A mis asesores, por ser mis guías durante el desarrollo de esta tesis. Gracias por sus consejos y por el tiempo que me han dedicado.

A mis amigas Claudia y Abigail, porque con ustedes he compartido los buenos y malos momentos. Sin ustedes no hubiera iniciado ni concluido este sueño.

A Sofía Reza, por su apoyo incondicional e inmensa ayuda a lo largo de estos años de maestría.

A todos los profesores de la sección de Computación, por todo el conocimiento que nos trasmiten y al CONACYT por el apoyo económico que me brindó.

Índice general

Indice de figuras xi

Indice de cuadros xiv

1. Introducción 1
- 1.1. Sistemas basados en conocimiento . . . 1
- 1.2. Verificación de sistemas basados en conocimiento . . . 3
- 1.3. Planteamiento del problema . . . 5
- 1.4. Propuesta de solución . . . 6
- 1.5. Objetivos . . . 7
  - 1.5.1. Objetivo general . . . 7
  - 1.5.2. Objetivos específicos . . . 7
- 1.6. Contribuciones . . . 8
- 1.7. Estructura del documento . . . 9

2. Antecedentes 11
- 2.1. Métodos de verificación . . . 11
  - 2.1.1. Métodos tabulares . . . 11
  - 2.1.2. Métodos basados en la generación de etiquetas . . . 12
  - 2.1.3. Métodos basados en grafos . . . 12
  - 2.1.4. Métodos basados en interpretaciones algebraicas . . . 13
  - 2.1.5. Métodos basados en meta - conocimiento . . . 13
  - 2.1.6. Métodos basados en redes de Petri . . . 13
- 2.2. Herramientas de verificación . . . 14
  - 2.2.1. EVA (Expert systems Validation Associate) . . . 14
  - 2.2.2. KB - REDUCER . . . 14
  - 2.2.3. COVER (COmpleteness VERifier) . . . 15
  - 2.2.4. IMVER (Incidence Matrix VERification) . . . 16
  - 2.2.5. VERITAS . . . 16
  - 2.2.6. CRIB . . . 17
  - 2.2.7. VALENS (VALid Engineering Support) . . . 17
  - 2.2.8. OAV - VVT . . . 17
  - 2.2.9. HESV . . . 18

2.2.10. VBCH . . . 18
2.3. Análisis comparativo y discusión . . . 19
2.4. Conclusiones . . . 22

3. Lenguaje de Representación del Conocimiento: HAries **23**
3.1. Características generales de HAries . . . 23
3.1.1. Componentes principales . . . 24
3.1.2. Estructuras de representación del conocimiento . . . 24
3.1.3. Razonamiento con incertidumbre . . . 25
3.2. Proposición . . . 25
3.2.1. Elementos de la proposición . . . 26
3.2.2. Sintaxis de las proposiciones . . . 28
3.2.3. Semántica de las proposiciones . . . 29
3.3. Regla de producción generalizada . . . 30
3.3.1. Elementos de la regla de producción generalizada . . . 30
3.3.2. Sintaxis de las reglas de producción generalizadas . . . 31
3.3.3. Semántica de las reglas de producción generalizadas . . . 32
3.4. Proceso evaluativo . . . 32
3.5. Conclusiones . . . 36

4. Proceso de Verificación **37**
4.1. Conceptos previos . . . 37
4.2. Clasificación y definición de anomalías . . . 39
4.3. Valores ilegales . . . 40
4.3.1. Métodos de detección de valores ilegales . . . 41
4.4. Redundancia . . . 41
4.4.1. Métodos de detección de redundancia . . . 43
4.5. Circularidad . . . 46
4.5.1. Métodos de detección de circularidad . . . 51
4.6. Inconsistencia . . . 53
4.6.1. Métodos de detección de inconsistencia . . . 55
4.7. Estudio empírico . . . 55
4.8. Conclusiones . . . 58

5. Análisis de Coeficientes **59**
5.1. Conceptos previos . . . 59
5.2. Clasificación y definición de coeficientes . . . 60
5.3. Coeficiente de mezcla . . . 61
5.3.1. Método para el cálculo del coeficiente de mezcla . . . 63
5.4. Coeficiente de profundidad . . . 63
5.4.1. Método para el cálculo del coeficiente de profundidad . . . 66
5.5. Índice de razonamiento global . . . 67
5.5.1. Método para el cálculo del índice de razonamiento global . . . 69

5.6. Índice de razonamiento directo 70
5.6.1. Método para el cálculo del índice de razonamiento directo 72
5.7. Estudio empírico 72
5.8. Conclusiones 74

6. Sistema VBCH **75**
6.1. Organización general del sistema 75
6.2. Módulo de extracción de parámetros 77
6.2.1. Especificación del módulo de extracción 77
6.2.2. Diseño de la interfaz del módulo de extracción 79
6.2.3. Ejemplo de aplicación del módulo de extracción 81
6.2.4. Estructura de clases del módulo de extracción 81
6.3. Módulo de verificación 84
6.3.1. Especificación del módulo de verificación 85
6.3.2. Diseño de la interfaz del módulo de verificación 86
6.3.3. Ejemplo de aplicación del módulo de verificación 87
6.3.4. Estructura de clases del módulo de verificación 89
6.4. Módulo de análisis de coeficientes 91
6.4.1. Especificación del módulo de análisis 92
6.4.2. Diseño de la interfaz del módulo de análisis 93
6.4.3. Ejemplo de aplicación del módulo de análisis 94
6.4.4. Estructura de clases del módulo de análisis 95
6.5. Conclusiones 98

7. Conclusiones **99**
7.1. Trabajos futuros 101

A. Función de contribución de una regla **103**

B. Función de combinación de reglas **105**

C. Proceso evaluativo en HAries **107**

Índice de figuras

1.1. Conceptos básicos del funcionamiento de un SBC. 2
1.2. Arquitectura del sistema VBCH. 7

3.1. Representación gráfica de una BC hipotética. 34
3.2. Incertidumbre en las reglas. 36

4.1. Clasificación de anomalías en HAries. 39
4.2. Anomalías estructurales en la base de reglas: Inclusión. 43
4.3. Anomalías estructurales en la base de reglas: Duplicación. 43
4.4. Anomalías estructurales en la base de reglas: Ciclo local. 46
4.5. Anomalías estructurales en la base de proposiciones: Ciclo local. 47
4.6. Anomalías estructurales en la base de reglas: Ciclo particular. 48
4.7. Anomalías estructurales en la base de proposiciones: Ciclo particular. . 49
4.8. Anomalías estructurales en la base de reglas y de proposiciones: Ciclo global. 51
4.9. Anomalías lógicas en la base de reglas: Contradicción. 54
4.10. Anomalías lógicas en la base de reglas: Condiciones innecesarias. 55

5.1. Clasificación de los coeficientes en HAries. 60
5.2. Cálculo del coeficiente de mezcla. 62
5.3. Cálculo del coeficiente de profundidad. 65
5.4. Cálculo del índice de razonamiento global. 68
5.5. Cálculo del índice de razonamiento directo. 71

6.1. Diagrama de contexto de VBCH. 76
6.2. Diagrama de contexto del módulo de extracción de parámetros. 77
6.3. Interfaz de usuario del módulo de extracción de parámetros. 80
6.4. Ejemplo de un reporte generado en el módulo de extracción. 81
6.5. Clases para la extracción de información. 82
6.6. Diagrama de contexto del módulo de verificación. 85
6.7. Interfaz de usuario del módulo de verificación. 87
6.8. Ejemplo de un reporte generado en el módulo de verificación. 88
6.9. Clases para la detección de anomalías. 90
6.10. Diagrama de contexto del módulo de análisis de coeficientes. 92
6.11. Interfaz de usuario del módulo de análisis de coeficientes. 93

6.12. Ejemplo de un reporte generado en el módulo de análisis de coeficientes. 94
6.13. Clases para el cálculo de coeficientes. 96

Índice de cuadros

2.1. Resumen de las herramientas de verificación 20

3.1. Proceso evaluativo para una proposición objetivo 35

4.1. Características de las BC 57
4.2. Ciclos de reloj en la ejecución de los métodos de detección de anomalías 57
4.3. Número de anomalías detectadas por caso de estudio 58

5.1. Características de las BC 73
5.2. Ciclos de reloj en la ejecución de los métodos para el cálculo de los coeficientes 73
5.3. Valores de los coeficientes por caso de estudio 73

6.1. Características que se reportan de cada estructura 78
6.2. Relaciones entre las estructuras de representación 78
6.3. Anomalías que se reportan de cada estructura 85

Capítulo 1

Introducción

1.1. Sistemas basados en conocimiento

La Inteligencia Artificial (IA) es una ciencia que trata de la comprensión de la inteligencia y del diseño de sistemas inteligentes, es decir, el estudio y la simulación de las actividades intelectuales del hombre (manipulación, razonamiento, percepción, aprendizaje, creación). Para esto, utiliza herramientas teóricas y experimentales de las ciencias de la computación y ha producido un cuerpo de principios, representaciones, algoritmos y nuevas tecnologías.

Existen múltiples técnicas que pueden ser programadas en una computadora con el fin de lograr un comportamiento inteligente, en base a ellas han surgido las llamadas disciplinas o ramas de la IA. En años recientes, estas técnicas han abordado muchos campos prácticos y desarrollado herramientas sofisticadas de uso cotidiano en sectores comerciales e industriales. Una de estas ramas que ha resultado muy eficaz para proporcionar competitividad en múltiples sectores son los *Sistemas Basados en Conocimiento* (SBC).

El profesor Edward Feigenbaum, pionero en la tecnología de los SBC, los ha definido como programas de computación inteligentes que usan el conocimiento y los procedimientos de inferencia para resolver problemas que son lo suficientemente difíciles como para requerir significativa experiencia humana para su solución [2]. Es decir, un SBC es un sistema de cómputo que contiene la experiencia, conocimiento y habilidad propios de una persona o grupo de personas especialistas en un área particular del conocimiento humano, de modo que permitan solucionar problemas específicos de dicha área de manera inteligente y satisfactoria, en forma de diagnósticos, instrucciones, predicciones o consejos ante las situaciones reales que se planteen. En los problemas algunos elementos de la solución, o todos, son desconocidos, por lo que determinar exactamente la meta que se debe alcanzar forma parte del problema. Normalmente estos problemas son resueltos por expertos humanos [3].

Los SBC proporcionan consejos y apoyo en la toma de decisiones, ofrecen opiniones informadas y pueden explicar su razonamiento. Además, permiten la manipulación de grandes volúmenes de información, que puede estar formada por: hechos, reglas, heurísticas e incluso modelos de predicción con grados de certeza.

Una característica adicional de los SBC, que muchos consideran fundamental, es la capacidad que poseen de justificar su propia línea de razonamiento. De esta forma, pueden explicar las consideraciones que conducen a una conclusión y el por qué de las preguntas, lo que aumenta la confianza de haber tomado la decisión correcta en la resolución de un problema en particular. El estilo adoptado para alcanzar estas características es la programación basada en reglas.

En la figura 1.1 se ilustra el concepto básico de un SBC. El usuario aporta los hechos o información al sistema y recibe consejos o experiencia como respuestas. En su interior, el SBC incluye sus dos componentes principales: la base de conocimiento y el mecanismo de inferencia.

La Base de Conocimiento (BC) contiene, codificado en algún lenguaje de representación, el conocimiento que se tiene de la materia, en ella se almacenan continuamente todas las experiencias accesibles para el sistema. El mecanismo o *motor de inferencia* emplea el conocimiento acumulado en la base para razonar y determinar cómo resolver un problema particular, y está diseñado para tomar decisiones y juicios basados en la misma [4].

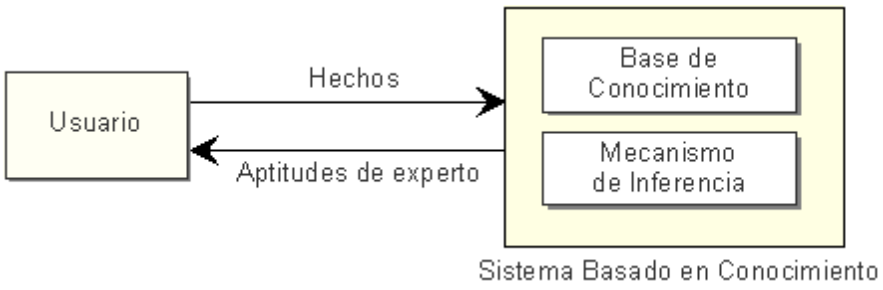

Figura 1.1: Conceptos básicos del funcionamiento de un SBC.

Además de la BC y el motor de inferencia, los SBC poseen:

- Módulo de adquisición del conocimiento: Mantiene la consistencia de la BC, puede incluir un editor y corregir errores sintácticos.
- Base de hechos: Es una memoria de trabajo que contiene en cada momento toda la información relativa al problema que se está resolviendo.
- Módulo de explicación: Permite al programa explicar su funcionamiento al usuario, indicando los pasos que han llevado a dar la solución propuesta.

- Interfaz de usuario: Permite la comunicación entre el usuario y el SBC.

El punto de partida de los SBC es la búsqueda de estructuras, procedimientos, métodos, etc., que permitan almacenar la experiencia acumulada sobre un tema dado, para poder hacer uso del mismo posteriormente en la solución de problemas en un dominio determinado. Los trabajos en esta dirección, parten de la definición de determinadas estructuras con características especiales, que permiten expresar situaciones del mundo real y su almacenamiento en las computadoras, a éstas se les denomina *Estructuras de Representación del Conocimiento* [5] y expresan, de alguna forma, los elementos de la BC.

Se han concebido varias estructuras para representar el conocimiento. Entre éstas se incluyen, reglas de producción, redes semánticas, marcos y otros, aunque también se ha utilizado lógica simbólica con este fin [6]. Las estructuras se pueden utilizar de manera individual o combinada (representación híbrida). Por ejemplo, se combinan las reglas con prototipos aprovechando las abstracciones de marcos dentro de reglas, marcos y redes semánticas formando redes de prototipo, etc. En consecuencia, han surgido *lenguajes de representación del conocimiento* [2] que aprovechan las diversas propiedades de las estructuras para complementar sus capacidades.

Una vez que el conocimiento ha sido almacenado en la base a través de las estructuras de representación, es necesario encontrar los mecanismos que permitan utilizar dicho conocimiento para resolver problemas, es decir construir el motor de inferencia. Existen dos formas básicas de construir un motor de inferencia: con encadenamiento hacia delante o hacia atrás. El encadenamiento hacia delante es el razonamiento que va de los hechos a las conclusiones que se desprenden de ellos, mientras que el encadenamiento hacia atrás comienza con un objetivo y pide información para confirmarlo o negarlo.

Es útil visualizar el encadenamiento hacia delante y hacia atrás como una ruta a través del espacio del problema en la que los estados intermedios corresponden a hipótesis intermedias bajo encadenamiento hacia atrás o conclusiones intermedias bajo encadenamiento hacia delante [2].

1.2. Verificación de sistemas basados en conocimiento

Desde principios de la década de los 80 los SBC están siendo frecuentemente usados en diferentes áreas como la industria, la ciencia, la defensa, la educación, etc., muchos de ellos en aplicaciones extensas y complejas.

La construcción de SBC complejos requiere modelar y representar grandes cantidades de conocimiento. Cuando se construye este tipo de sistemas, el experto da una

lista de reglas, hechos y restricciones. Se supone que el conocimiento del experto no conduce a ninguna contradicción; sin embargo, los problemas estructurales aparecen en la mayoría de los SBC. Por lo cual últimamente se ha prestado mucha atención en la verificación de los mismos.

"El principal objetivo del proceso de verificación es confirmar que el sistema es lógicamente consistente, aunque esto no garantiza que su conocimiento dependiente del dominio corresponda con el del experto humano" [7].

Durante la verificación se analizan propiedades bien definidas de un SBC. Dependiendo del tipo de propiedades, la verificación puede estar restringida a una parte específica del sistema, como su base de conocimientos, su motor de inferencia o su interfaz de usuario, y puede también estar enfocada en aspectos específicos de la funcionalidad del sistema como su comportamiento deductivo [8].

A pesar de que las definiciones de verificación de BC varían en la literatura, es común a todas que el análisis incluya el chequeo de las estructuras que la integran para detectar anomalías lógicas, semánticas y estructurales introducidas durante las etapas de diseño e implementación. En adición, el término también implica otras metas tales como asegurar el mantenimiento, la seguridad y el servicio del sistema.

Existen tres propiedades formales principales que tienen que ver con la verificación de la BC, éstas son: consistencia, redundancia y circularidad. Por lo general estas propiedades surgen de la representación del conocimiento basado en reglas [8], aunque la forma en que se presentan depende de las características del lenguaje de representación del conocimiento que se esté utilizando.

- Una BC es *inconsistente* si puede producir resultados en conflicto de una entrada de datos válida. Conflicto implica contradicción lógica, como la deducción de un hecho y de su negación. La inconsistencia también incluye incompatibilidades semánticas.

- Una BC es *redundante* si contiene reglas que pueden ser eliminadas sin afectar el poder deductivo de la misma.

- Una BC presenta *circularidad* si contiene secuencias de reglas tales que la evaluación de éstas conduzcan a la formación de ciclos durante la ejecución del sistema (condiciones y acciones cruzadas entre pares o conjuntos de reglas)

La verificación también incluye la revisión de la sintaxis y de tipos (detectando valores y rangos ilegales, etc.)

Las anomalías mencionadas pueden provocar en el sistema diferentes consecuencias. Por ejemplo la presencia de redundancia en reglas degrada la ejecución del sistema y

además puede originar problemas cuando se trate de extenderlo, refinarlo o simplemente darle mantenimiento. Mientras que las secuencias circulares de reglas provocan ciclos infinitos, puesto que el sistema de inferencia no los reconoce en tiempo de ejecución.

El despliegue acertado de la tecnología del conocimiento, depende de un gravamen favorable de las ventajas sobre los riesgos estimados. Debe ser demostrado que las ventajas probables del uso compensan los riesgos cuando los SBC se utilizan en áreas críticas, como por ejemplo el diagnóstico médico. Por lo tanto, es prudente establecer qué tan seguro y eficaz es un sistema antes de que se confíe en él para tomar decisiones automáticas.

1.3. Planteamiento del problema

Cuando se construyen SBC, es usual que la información llevada a la base esté integrada por grandes cantidades de conocimiento. Esto puede conducir a la introducción de múltiples anomalías y errores durante la descripción de las estructuras que la conforman. Regularmente la detección de estos problemas resulta muy difícil e implica grandes gastos de tiempo, sobre todo para desarrolladores de poca experiencia.

Esta dificultad se hace aún más compleja cuando se emplean esquemas híbridos para representar el conocimiento, puesto que se pueden establecer relaciones entre las diversas estructuras que define el lenguaje, cuyas diversas componentes se construyen, por lo general, de forma independiente y en su conjunto pueden provocar problemas globales.

Otro inconveniente que se presenta durante la concepción de la BC es que no existe una forma directa durante el proceso de desarrollo, de evaluar el conocimiento para alertar sobre posibles deficiencias en cuanto a cantidad y organización. Como complemento, resulta muy conveniente contar con ciertas medidas para evaluar de forma global o quizás específica, la configuración del conocimiento representado, mediante información cuantitativa sobre las estructuras internas de la BC.

Los métodos y herramientas de verificación buscan ayudar en la solución de estos problemas y al mismo tiempo, la verificación es una parte importante en el aseguramiento de la confiabilidad de los SBC.

Para constituir un procedimiento de verificación se debe partir de la identificación y definición de las anomalías que se pueden presentar, tomando en cuenta la declaración de las estructuras que posee el lenguaje de representación que se esté utilizando para construir la BC. Posteriormente se debe especificar un modelo semántico que permita formalizar y automatizar el método de verificación mediante un proceso eficiente de inferencia.

1.4. Propuesta de solución

Las herramientas de verificación han sido desarrolladas usualmente para sistemas que utilizan esquemas únicos de representación del conocimiento o sistemas en los cuales no es posible definir relaciones entre sus estructuras. En este trabajo, tomando como base los problemas que se describen en la sección anterior, se plantea el desarrollo de un sistema de verificación y análisis de BC que posean una representación híbrida del conocimiento (reglas y marcos).

De manera general el sistema propuesto, VBCH, divide el análisis en dos partes: (1) detección de las posibles anomalías en cada una de las estructuras almacenadas en la BC y (2) detección de situaciones conflictivas que se deriven de las relaciones que se establecen entre estas estructuras. De esta forma, permitirá confirmar que una BC es lógicamente consistente, tras la detección de errores lógicos, semánticos y estructurales.

El estudio se ha realizado considerando BC construidas con el lenguaje de representación del conocimiento HAries [1]. Por tanto, se podrán verificar de forma automática las BC de cualquiera de las aplicaciones prácticas desarrolladas con dicho lenguaje.

Por otra parte se podrán extraer características para el análisis de la estructura teórica de las bases, mediante la construcción de tablas de parámetros estadísticos simples y gráficos, que muestren la composición de las mismas. Además se propone el cálculo de un conjunto de coeficientes, que además de brindar datos sobre la organización, también vaya orientado a alertar sobre posibles deficiencias que pudieran afectar la ejecución exitosa de un sistema que trabaje con tales tipos de conocimientos.

VBCH está compuesta por tres módulos fundamentales:

1. Extracción de parámetros
2. Verificación
3. Análisis de coeficientes

El módulo de *extracción de parámetros* consiste en la construcción de tablas de parámetros estadísticos simples y gráficos. Brinda al programador información sobre la composición de las BC, como por ejemplo el estado de los archivos, el número total de cada estructura definida en el lenguaje e información específica de tales estructuras.

El módulo de *verificación* brinda algunas facilidades en el proceso de desarrollo de las aplicaciones mediante la detección de errores como: redundancia, circularidad, inconsistencia en reglas y valores ilegales.

El módulo de *análisis de coeficientes* se fundamenta en el cálculo de un conjunto de índices para la evaluación de la organización del conocimiento almacenado. En este

módulo se obtiene información cualitativa sobre las estructuras utilizadas en la BC y brindará orientación con el fin de detectar posibles deficiencias en la representación del conocimiento empleada.

En la figura 1.2 se muestra la arquitectura general del sistema, organizada por niveles. En el nivel más bajo se encuentra el sistema de adquisición que provee el lenguaje, formado por un conjunto de editores y compiladores para cada estructura de representación. En el siguiente nivel se encuentra la BC, que almacena el conocimiento incorporado en el nivel anterior. Sobre la base se halla el módulo de extracción que obtiene información sobre las estructuras y sus relaciones. Esta exploración además de ser manipulada para la elaboración de un reporte sobre la composición de la BC, se emplea en las próximas capas para efectuar la verificación y el cálculo de coeficientes.

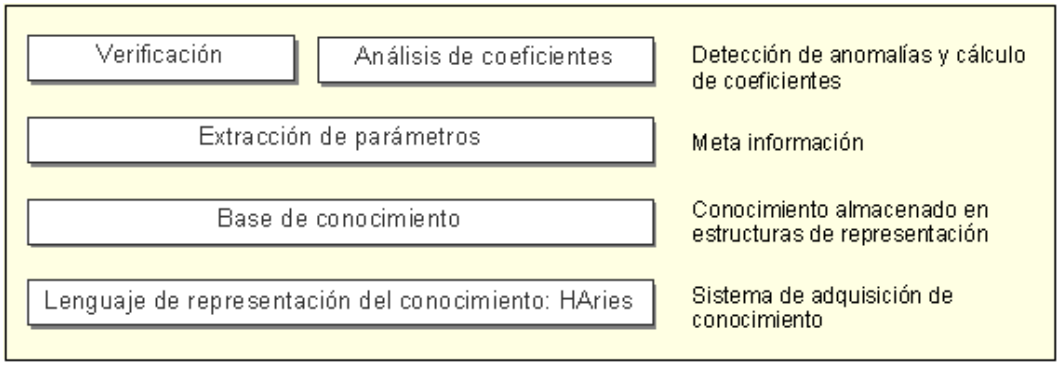

Figura 1.2: Arquitectura del sistema VBCH.

1.5. Objetivos

Para dar solución a las necesidades planteadas y poder cumplir satisfactoriamente con cada etapa de investigación y desarrollo de la propuesta descrita anteriormente, se han definido un conjunto de objetivos que se relacionan a continuación.

1.5.1. Objetivo general

Desarrollar un sistema que permita analizar y verificar la estructura teórica de Bases de Conocimientos, descritas con el lenguaje de representación HAries.

1.5.2. Objetivos específicos

1. Desarrollar un módulo para la extracción de parámetros generales y particulares de las estructuras que conforman la BC de tipo HAries.

2. Identificar y definir formalmente las anomalías locales y globales (derivadas de la relaciones) para cada una de las estructuras de representación del conocimiento que especifica el lenguaje.

3. Clasificar los errores definidos y crear métodos de verificación que se puedan aplicar para la detección de los mismos a cada estructura.

4. Implementar un modelo que permita integrar y automatizar los métodos de verificación mediante un proceso eficiente de inferencia.

5. Definir un conjunto de coeficientes e índices para evaluar la calidad del conocimiento almacenado en una BC y brindar una interpretación de los valores que pueden tomar.

1.6. Contribuciones

El sistema de verificación y análisis VBCH ha sido concebido como una extensión al lenguaje de representación del conocimiento HAries. VBCH en primera instancia permite la interpretación de las estructuras empleadas para almacenar información en las BC descritas con este lenguaje. Partiendo de esta interpretación, se pueden aplicar diversos algoritmos para la detección automática de anomalías y métodos para el cálculo de medidas enfocadas a la evaluación de la estructura de la BC.

Los algoritmos de verificación propuestos pueden ser aplicados a bases construidas con cualquier lenguaje que utilice representación basada en reglas y con nociones de marcos; siempre y cuando se creen los métodos para extraer la meta-información necesaria y ésta sea proporcionada como entrada en dichos algoritmos.

De la misma forma ocurre con los coeficientes para el estudio de la composición de las BC, ya que estas mesuras y sus interpretaciones se pueden ajustar de forma intuitiva para ser aplicadas a cualquier base de reglas (representación más comúnmente empleada en la construcción de SBC).

Desde el punto de vista de los usuarios del lenguaje HAries, las aportaciones más precisas tienen que ver directamente con las facilidades que brinda la herramienta para la localización de las anomalías y la reducción de los tiempos de detección de las mismas, sobre todo para desarrolladores de poca experiencia. Para esto, el proceso de verificación está dividido por el tipo de problema y por la estructura de representación a analizar. Además, el sistema posee la característica de limitar la búsqueda de errores a un rango específico de elementos, brindando de este modo resultados precisos cuando el usuario tiene conciencia de la posible ubicación de la falta.

En lo referente al análisis de la BC, se proveen patrones que permiten al usuario percatarse de posibles insuficiencias al momento de representar el conocimiento, una vez

que éste ha sido adquirido a través de las percepciones de expertos en la materia, libros u otro medio. Está claro que los resultados que brinda este análisis no son definitivos en cuanto a la correctitud de la BC, simplemente se obtiene la información que debe hacer reflexionar sobre la estructura que se ha construido. No se debe olvidar que en muchos problemas puede ocurrir, que el nivel de los conocimientos existentes no permita la elaboración de un sistema con grandes procesos de razonamiento, pero también que la detección y representación de dichos conocimientos, es difícil y constituye la base fundamental del éxito en sistemas de este tipo.

1.7. Estructura del documento

Este documento de tesis se encuentra dividido en 7 capítulos. En el segundo capítulo se presentan aspectos relacionados con las bases teóricas del trabajo, respecto a los métodos de verificación más importantes, así como un resumen y comparación de algunas herramientas que implementan dichos métodos.

En el capítulo tres se muestran las características principales del lenguaje HAries y los elementos más importantes de las estructuras que posee para representar el conocimiento, haciendo énfasis en la sintaxis, el proceso de inferencia y la estrategia de control que implementa el ambiente de programación.

El capítulo cuatro introduce los conceptos conexos con los problemas que puede detectar el sistema VBCH durante el proceso de verificación. Este capítulo inicia con la definición formal de una BC descrita con HAries y tomando esto como base continúa con la clasificación y definición de las anomalías.

En el capítulo cinco se definen los coeficientes para el análisis de la BC, su necesidad y los métodos que utilizan para ser calculados. Los conceptos que se emplean para estos fines se especifican también en este apartado. Al mismo tiempo, la explicación de cada coeficiente va apoyada con un ejemplo para una mejor comprensión.

El capítulo seis presenta el diseño del sistema, desde el punto de vista de su arquitectura de software, y la implementación del mismo. En la etapa de diseño se puntualiza acerca de la organización general del sistema, la función y algoritmos de los módulos principales y finalmente las interfaces de usuario para cada uno de ellos. En la fase de implementación se exponen las jerarquías de clases, la explicación de los métodos que efectúan y por último se ejemplifica, mediante un caso de estudio, el funcionamiento de la herramienta.

La discusión de los resultados obtenidos y las conclusiones se brindan en el capítulo siete. Además se identifican algunas necesidades no resueltas y se presentan elementos que pueden ser incorporados para dar continuidad al sistema como trabajo futuro.

Los apéndices contienen aspectos particulares relacionados con el proceso evaluativo del lenguaje de representación del conocimiento empleado.

Capítulo 2

Antecedentes

Para responder al importante crecimiento de los sistemas basados en conocimiento y con el fin de asistir en la construcción de los mismos, muchos métodos y herramientas de verificación han sido desarrollados. Sin embargo, la mayoría de éstos que detectan anomalías en el contenido de la BC se aplican solamente a tal base, sin tomar en cuenta los aspectos deductivos y de control; a pesar de que muchas veces, la exactitud de los resultados obtenidos dependerá de las características del motor de inferencia [9].

En este capítulo se presentan los métodos más relevantes y algunas herramientas de verificación con sus características principales.

2.1. Métodos de verificación

Los métodos de verificación pueden ser agrupados en diferentes familias, de acuerdo al enfoque que siguen [9], [10]. Se han revisado las siguientes familias:

- Métodos tabulares
- Métodos basados en la generación de etiquetas
- Métodos basados en grafos
- Métodos basados en interpretaciones algebraicas
- Métodos basados en meta - conocimiento
- Métodos basados en redes de Petri

2.1.1. Métodos tabulares

Los métodos tabulares fueron los primeros en ser diseñados para detectar anomalías en BC. La estrategia que siguen estos métodos consiste en ir comparando pares de reglas de la base de reglas, con el objetivo de descubrir ciertas relaciones entre sus premisas

y conclusiones. Dado que los métodos tabulares sólo comparan pares de reglas, todas las inconsistencias que involucran más de dos reglas quedan fuera de su alcance [11].

Las técnicas más usadas en este tipo de métodos son las tablas de decisión. Éstas reúnen tradicionalmente las facilidades de representación del conocimiento con capacidades de comprobación adicionales como completitud, consistencia y redundancia.

La representación de la tabla de decisión está caracterizada por la separación entre las condiciones y las acciones, por una parte, y las expresiones condicionales (estados), por otra. Cada columna de la tabla (columna de la decisión) indica qué acciones deben o no ser ejecutadas para una combinación específica de estados en la condición. Los algoritmos entonces examinan la existencia de relaciones entre filas y columnas.

La desventaja de este método es que es útil solamente para bases de reglas pequeñas, por las proporciones de la tabla.

2.1.2. Métodos basados en la generación de etiquetas

Estos métodos han crecido con el diseño de los sistemas de mantenimiento de verdad (ATMS) por Kleer [5]. Consiste en la propagación de restricciones, especificadas por un conjunto de variables y condiciones en subconjuntos de los valores de las variables. El proceso va asignando valores a estas variables incrementalmente y chequeando sucesivamente la consistencia.

Dado que muchas veces la generación y propagación de todas las posibles bases de hechos iniciales es computacionalmente intratable, estos métodos toman el concepto de contexto, usado en el ATMS, como una forma de especificar un conjunto de bases de hechos iniciales. Los contextos que describan bases que causen la deducción de inconsistencias serán especialmente interesantes. Por lo tanto, la meta de estos métodos será la construcción de aquellos contextos. Como el número de reglas en la base de reglas es finito, las especificaciones contenidas en un contexto también serán finitas, aún si el número de posibles bases de hechos iniciales es infinito [12].

2.1.3. Métodos basados en grafos

Esta familia de métodos representa los sistemas basados en reglas (SBR) mediante grafos [23]. Los grafos son herramientas muy atractivas para representar dependencias conceptuales. Más aún, la teoría de grafos provee técnicas de análisis para estudiar propiedades como la conectividad o alcance en forma rigurosa y formal.

Las siguientes son algunas de las técnicas que se basan en este método:

Grafos dirigidos: Estos fueron ideados para detectar errores estructurales en SBR. Modela los conjuntos de reglas usando grafos dirigidos, con literales y reglas como

nodos y las asociaciones entre ellas como arcos. La verificación localiza errores de redundancia, conflicto, circularidad y metas inalcanzables. La detección se realiza con el análisis del alcance de trayectorias en el grafo dirigido [13].

Transformación algebraica de grafos: Esta técnica se concentra sobre todo en la redundancia y la inclusión de conocimiento estructural en una base de reglas. Emplea hipergrafos para alcanzar esto, representando las proposiciones antecedentes y sucedentes como nodos y las reglas como hiperarcos. Esta aproximación implica un tratamiento extremadamente exacto y formal del tema, incluyendo el desarrollo de una notación formal para los hipergrafos y el uso de una gramática y lenguaje gráficos. El ejercicio también se amplía a los sistemas de reglas que implican incertidumbre [8].

2.1.4. Métodos basados en interpretaciones algebraicas

Éstos son sistemas basados en la transformación de la BC en una estructura algebraica, como un álgebra booleana y la aplicación de conceptos y procedimientos relativos a esa estructura algebraica para la verificación de la BC. Esta familia de métodos son los más confiables, puesto que se basan sólidamente en conceptos algebraicos. Sin embargo, sólo pueden ser aplicados en bases de reglas proposicionales [9].

2.1.5. Métodos basados en meta - conocimiento

Con el fin de verificar la coherencia de grandes bases de reglas, el objetivo primordial de estos métodos es encontrar maneras de reducir la combinatoria mientras que intentan encontrar tantas anomalías como sea posible. Esto se logra gracias al uso de meta-conocimiento. Primero, permiten describir características significativas de conceptos y reglas usando meta atributos, lo cual reduce la cantidad de conocimiento referida mientras se comparan dos reglas. Además, algunos atributos ligando pedazos de conocimiento constituyen los caminos de acceso permitiendo reducir la duración de la búsqueda de una información particular. Permiten también guardar el rastro de los resultados de los tratamientos previamente realizados que serán útiles en el proceso futuro, para evitar repetir tratamientos innecesarios [14].

2.1.6. Métodos basados en redes de Petri

Los métodos en esta familia están basados en el modelado de la BC usando redes de Petri y la aplicación de técnicas en esta área con el fin de detectar inconsistencias y otros errores semánticos. Estos métodos necesitan probar el modelo bajo todos los posibles estados iniciales para garantizar la ausencia de inconsistencias, lo cual puede ser computacionalmente muy costoso [11].

Una de las técnicas que se basan en este método es la red de Petri con matriz de incidencia. Esta aproximación busca detectar errores estructurales en SBR sin incertidumbre. La verificación cubre errores de redundancia, inconsistencia, incompletitud y

circularidad. El SBR se modela como una red de Petri donde los literales están representados por lugares y las reglas por transiciones. La red de Petri entonces se transforma en una matriz de incidencia usando las transiciones como filas y los lugares como columnas. La detección es desarrollada de manera incremental, por multiplicación de la matriz de incidencia por un vector de representación de las nuevas reglas [15].

2.2. Herramientas de verificación

Existen numerosas herramientas de verificación; éstas pueden dividirse en dos categorías: dependientes e independientes del dominio [9]. Las herramientas dependientes usan información específica acerca del dominio en particular de la BC para verificar la validez del conocimiento, mientras las independientes tratan de detectar anomalías que consisten en un abuso o uso inusual de los esquemas de representación del conocimiento.

Enseguida se enumeran algunas de las herramientas que han sido reportadas.

2.2.1. EVA (Expert systems Validation Associate)

Es una familia de herramientas desarrolladas para verificar aplicaciones escritas en diversos *shells* de sistemas expertos. El objetivo del proyecto EVA fue ambicioso: construir un conjunto de herramientas integradas para chequear redundancia, consistencia y completitud de cualquier SBC escrito en cualquier lenguaje de representación del conocimiento [16]. Sin embargo las metas planteadas sólo se resolvieron en forma limitada.

Se seleccionó la lógica de primer orden como formalismo de representación del conocimiento, y Prolog como plataforma de implementación del conjunto de herramientas. Las definiciones de anomalías en la BC usadas por EVA cubren la representación del conocimiento mediante reglas y los aspectos declarativos de los marcos.

Cada herramienta de chequeo en EVA posee dos formas: básica y extendida. La diferencia entre ellas radica en la incorporación de meta-conocimiento; esto permite la resolución de conflictos por incompatibilidad y la detección de aserciones simultáneas de instancias de los términos involucrados [16].

2.2.2. KB - REDUCER

La primera versión de KB-Reducer (KBR1) estuvo motivada por dos requerimientos: proveer automáticamente chequeo de consistencia y redundancia en BC y una forma compilada apropiada para las mismas [16]. Fue desarrollado para chequear SBC escritos en lógica proposicional. El algoritmo usado está basado en el diseño de los ATMS, en los cuales se encuentra el conjunto de suposiciones que se deben sostener para que cada hipótesis de la BC sea verdadera. Estos conjuntos son llamados etiquetas

para la hipótesis.

El algoritmo de KB-Reducer requiere que la BC pueda ser estratificada en niveles, de acuerdo a las dependencias de inferencia en las reglas. El algoritmo trabaja procesando cada regla determinando la etiqueta de la misma. KB-Reducer realiza el chequeo de redundancia inmediatamente después de computar cada etiqueta. Para informar cualquier anomalía al usuario, el KB-Reducer necesita registrar las identidades de las reglas usadas para crear las etiquetas, para examinar las cadenas y las reglas problemáticas de inferencia.

Para que KB-Reducer produzca resultados significativos, el motor de inferencia debe ser monotónico, no-selectivo (ninguna estrategia de resolución de conflictos), y guiado por los datos (encadenamiento hacia delante). KBR2 fue una extensión de KBR1 para verificar BC escritas en un subconjunto de lógica de primer orden y KBR3 para comprobar BC con ecuaciones [16].

2.2.3. COVER (COmpleteness VERifier)

La primera versión de COVER fue usada para verificar una aplicación médica de SBC. Al igual que EVA, COVER está basado en Prolog, y realiza la verificación usando una estrategia de prueba de teoremas dirigida por metas. Uno de los objetivos de este proyecto fue definir formalmente un conjunto comprensivo de anomalías en una BC [17]. Estas definiciones formales son usadas como base de comparación de varias de las herramientas de verificación.

Las operaciones de verificación de COVER se dividen en tres grupos: operaciones de comprobación de integridad, de reglas y de cadenas de reglas. La razón para esta separación sólo tiene que ver con el costo de cómputo en los chequeos de cada caso.

El inspector de integridad construye varias tablas, incluyendo una que contenga todos los datos contra los valores que pueden tomar legalmente (mediante una regla o pregunta al usuario). Una vez completada, la tabla se utiliza en la comprobación de condiciones que no se satisfacen, reglas de punto muerto y valores faltantes. El inspector de reglas compara los antecedentes y sucedentes de cada par de reglas. La implementación del inspector de cadenas de reglas se divide en tres sub-procedimientos: crear ambientes de meta, comprobar estos ambientes para saber si hay redundancia y finalmente comprobar los mismos para saber si hay deficiencia. El usuario debe primero seleccionar la meta o el sistema de metas a ser comprobado. Después COVER usa un meta-intérprete similar a un motor de inferencia de encadenamiento hacia atrás para realizar los chequeos [17].

2.2.4. IMVER (Incidence Matrix VERification)

El sistema experimental de verificación IMVER-1 emplea la técnica de matriz de incidencia, cuyos elementos representan las proposiciones que pueden estar presentes en una regla. Estas matrices se analizan para establecer la existencia (o no-existencia) de los errores y anomalías en la base de reglas.

Si una proposición particular aparece en una regla, la intersección apropiada de regla/proposición (renglón/columna) en esa posición de la matriz se indica con un 1. En caso contrario las intersecciones serán puestas en 0. Para implementar la verificación cada regla en la base debe ser comparada con cada una de las otras. Las anomalías tratadas por IMVER son: inclusión, duplicación, inconsistencia y circularidad [18].

Las críticas principales dirigidas a este sistema fueron: la cantidad considerable de espacio requerido para almacenar las matrices y la energía de proceso necesitada para realizar las multiplicaciones requeridas de la matriz y las comparaciones necesarias. Estas críticas no son sólo aplicables al sistema IMVER-1, sino a todos los sistemas de verificación que utilicen técnicas de matriz de incidencia y fueron la razón por la cual se comenzó con el diseño del sistema IMVER-2 que utiliza una representación codificada binaria avanzada [18]. Las ventajas ofrecidas por esta representación son: un ahorro considerable del espacio y una reducción significativa en el proceso de búsqueda errores y anomalías en la BC.

2.2.5. VERITAS

VERITAS es una herramienta para la verificación automática de BC [19]. Esto lo consigue realizando un análisis estructural que permite la detección de las anomalías. Aunque inicialmente se empleó para verificar la base SPARCE, la herramienta fue desarrollada con propósitos genéricos de verificación, es independiente del dominio del conocimiento y de la gramática de las reglas.

Su arquitectura es abierta y modular permitiendo la interacción del usuario a lo largo de todo el proceso de la verificación. Puesto que la herramienta es independiente de la gramática de la BC, cualquier SBR se puede analizar teóricamente por VERITAS. Está dividida en cuatro módulos principales: un convertidor, un módulo de administración de una base de datos interna, un editor de reglas y herramientas de verificación.

El módulo del convertidor permite la representación de reglas externas en una forma canónica que es reconocida por el resto de los módulos y el de administración de la BD interna es responsable de la extracción y de la clasificación de toda la información necesitada durante la fase de detección de anomalías. Las herramientas de verificación tratan anomalías de redundancia y circularidad [19].

2.2.6. CRIB

CRIB es un sistema que permite detectar inconsistencias definidas por restricciones de integridad en una BC [9]. Esta herramienta permite verificar bases expresadas en un lenguaje de representación del conocimiento llamado CCR-2.

Además incorpora un lenguaje para la especificación de éstas restricciones, lo cual permite al usuario no sólo analizar un conjunto dado de las mismas, sino también modificarlas o eliminarlas y estudiar los efectos de este cambio, sin necesidad de repetir todo el proceso de creación [9]. Se ingresa también, como información adicional para ayudar al usuario a detectar las malformaciones, nombre y estado (no aplicable, satisfecha, violada) para cada una de las restricciones. Las entradas del sistema son la BC y un conjunto de restricciones de integridad. La salida proporciona las trayectorias deductivas que conducen a la violación de cierta restricción.

CRIB no considera el esquema de la base de hechos. Este esquema se puede definir como el dominio del problema a solucionar, no el problema particular. El esquema de una base de hechos de CCR-2 define aspectos tales como la taxonomía del marco, las relaciones, las proposiciones, etc.

2.2.7. VALENS (VALid Engineering Support)

Es un sistema que permite detectar anomalías en BC construidas con el lenguaje Aion, un ambiente de desarrollo de sistemas basados en reglas. VALENS puede ser usado durante o después de la construcción de la BC y se enfoca en la detección de contradicciones lógicas, circularidad entre pares de reglas, completitud y redundancia. Utiliza meta-información para realizar la verificación y está escrito en el mismo lenguaje que las bases que analiza: Aion. Las entradas de VALENS son la BC escrita en Aion y la selección de las reglas a verificar. La salida es un documento en el cual son reportadas todas las reglas (combinaciones) inválidas detectadas [20].

El algoritmo de verificación que usa VALENS realiza tres pasos principales: construcción de un meta modelo, selección de las anomalías potenciales y finalmente la detección de las anomalías elegidas. Durante la construcción del meta modelo se analizan las reglas que conforman la base y se dividen en condiciones y conclusiones, las condiciones a su vez son divididas en cláusulas. Además la meta información incluye los valores máximos y mínimos de las variables y de los atributos.

2.2.8. OAV - VVT

OAV-VVT es un sistema de verificación que puede ser aplicado a varios esquemas de representación del conocimiento (redes semánticas, marcos y reglas de producción) después de aplicarles una transformación para obtener una base con representación de tripleta Objeto - Atributo - Valor, llamada Ex-OAV. En esta fase de transformación, se

cambia la sintaxis de la representación pero la semántica permanece inalterada. Todo el conocimiento en la BC Ex-OAV es representado por un sistema basado en reglas usando tripletas [21].

La nueva base obtenida es la entrada en la herramienta de verificación, donde se ejecuta la detección de circularidad y redundancia en reglas, conjuntamente con el análisis de completitud y consistencia. Este proceso es reiterativo y todo el conocimiento no válido es reportado. OAV-VVT puede usarse durante la fase de representación de conocimiento o refinamiento de la BC.

2.2.9. HESV

Es un sistema que permite detectar anomalías en BC definidas con el lenguaje HES usando redes de Petri coloreadas y el concepto de tokens de estado controlados. La red de Petri modela sistemas expertos híbridos (basados en reglas y marcos) y se definen anomalías adicionales causadas por la integración de jerarquías de objeto y las reglas de producción [22].

El descubrimiento y análisis de las anomalías del modelo usado en esta herramienta, son hechos construyendo y examinando el árbol de inferencia de conocimiento, proporcionando una comprobación de la consistencia de la base de conocimiento híbrida. Las entradas de HESV son la BC y la selección de las anomalías a tratar. La salida proporciona una representación gráfica del sistema y las reglas involucradas en la anomalía.

2.2.10. VBCH

Es una herramienta que permite la verificación de bases de conocimiento descritas con el lenguaje de representación HAries [23]. VBCH utiliza una combinación de las técnicas de meta-conocimiento y grafos dirigidos.

En su primera etapa obtiene toda la meta información que será empleada durante la detección de errores. A esta etapa se le ha denominado de extracción de parámetros y constituye el primer módulo del sistema VBCH. La información incluye la separación de los antecedentes (condiciones) y sucedentes (acciones) de las reglas, la división en partes (cláusulas) de los antecedentes, una lista con todas las reglas en las que aparece una acción con parte del sucedente, los valores máximos de las estructuras, etc.

La segunda etapa se encarga de la localización de las anomalías y está materializada en el módulo de verificación. Aquí se emplean los grafos dirigidos al crear el árbol de inferencias para cada acción en el sucedente de una regla. Los árboles son recorridos para determinar si existen ciclos partiendo de una acción específica (circularidad), y comparados para determinar si existen coincidencias (redundancia o contradicciones lógicas en reglas). Cada nodo en el árbol representa una proposición (puede ser condición o acción en dependencia de la posición que ocupa en la regla) y los arcos son las

reglas que las asocian.

La tercera etapa se encarga del estudio de la base de reglas y también emplea la fase de extracción de parámetros con este fin. Ésta se concreta en el módulo de análisis de coeficientes y del mismo modo utiliza grafos dirigidos, aunque la constitución del grafo depende de la medida que se esté calculando.

2.3. Análisis comparativo y discusión

El estudio anterior permite establecer una comparación entre las herramientas investigadas y VBCH; tomando en consideración aspectos relacionados con el lenguaje de representación utilizado para construir las BC que verifican y las anomalías que detectan.

Los siguientes son los parámetros que se eligieron para establecer la comparación:

1. Lenguaje de representación del conocimiento
2. Método de verificación que utilizan
3. Representación del conocimiento
   - BR: Basado en reglas
   - BM: Basado en marcos
   - OAV: Tripletas Objeto - Valor - Atributo
4. Estrategia de encadenamiento
   - ED: Encadenamiento hacia delante
   - EA: Encadenamiento hacia atrás
5. Razonamiento aproximado (FC: Factores de certidumbre)
6. Anomalías tratadas
   - CI: Circularidad
   - CO: Completitud
   - IC: Inconsistencia
   - RE: Redundancia
7. Implementación

Herramienta	1	2	3	4	5	6	7
EVA	Diversos *shells*	Meta - conocimiento	BR, BM	ED	FC	CO, IC, RE	Prolog
KB - REDUCER	Lenguajes basados en lógica proposicional	Generación de etiquetas	BR, OAV	ED	-	RE	Lisp
COVER	Prolog	Generación de etiquetas	BR, BM	EA	-	CO, RE	Prolog y C
IMVER	IMVER	Tabulares	BR	ED, EA	-	CI, IC, RE	C
VERITAS	SPARCE	Transformaciones algebraicas	BR	EA	-	CI, RE	Prolog y C
CRIB	CCR-2	Generación de etiquetas	BR, BM	EA	FC	IC	?
VALENS	Aion	Meta - conocimiento	BR	EA	-	CI, CO, IC, RE	Aion
OAV - VVT	ExOAV	Grafos	BR, OAV	EA	-	CI, CO, IC, RE	Visual Prolog
HESV	HES	Redes de Petri	BR, BM	EA	-	CI, CO, IC, RE	?
VBCH	HAries	Meta - conocimiento, grafos	BR, BM	EA	FC	CI, IC, RE	Visual C++

Cuadro 2.1: Resumen de las herramientas de verificación

Como se mencionó en su descripción, VBCH utiliza como *método de verificación* una combinación de las técnicas de meta-conocimiento y grafos dirigidos. En la construcción del meta-modelo, durante la etapa de extracción, existe gran similitud con la herramienta VALENS. Ya en la fase de localización de anomalías utiliza grafos dirigidos, de la misma forma que en OAV-VVT.

La *representación del conocimiento* viene dada por las características del lenguaje que se emplea en la construcción de las BC que la herramienta de verificación va a analizar. Como se puede observar en el cuadro comparativo, la mayoría de los SBC se basan en la representación por medio de reglas de producción (BR) y casi en todos los casos incorporan nociones de marcos (BM), aunque éstas se presentan en formas diversas. Sólo KB-REDUCER y OAV-VVT emplean tripletas del tipo Objeto - Valor - Atributo (OAV).

En este punto VBCH presenta dos diferencias principales con el resto de las herramientas, dada la naturaleza de las reglas en el lenguaje HAries:

1. La composición de la parte condicional de la regla (antecedente) y
2. El número de acciones (sucedentes) de la regla y los factores de certidumbre asociados

En relación a la composición del antecedente, la mayoría de las herramientas estudiadas hace análisis de reglas cuyas condiciones están expresadas en forma normal conjuntiva; mientras que en HAries se permite introducir expresiones mucho más complejas utilizando de forma combinada los conectivos que define el lenguaje. Esto provoca que los métodos para la detección de las anomalías en reglas sean más complejos ya

que implican el análisis de la condición y la división en partes de las mismas, a través de los conectivos[1].

Por otra parte, HAries permite reunir en la misma regla varias acciones como consecuencia de la condición; y además incorpora el manejo de *factores de certidumbre* (FC) que indican una medida cuantitativa de la creencia acerca del cumplimiento o no de algún hecho dado, la cual no tiene que ser extremal (Si o No). Esto hace que el análisis del sucedente de la reglas también se dificulte puesto que hay que examinar cada sucedente y los valores de verdad asociados a los mismos. El manejo de certidumbre es incorporado solamente en las herramientas EVA y CRIB de todas las estudiadas.

La *estrategia de encadenamiento* es muy importante en la detección de anomalías, pues posee relación con el proceso de inferencia. Por ejemplo, los algoritmos de detección de circularidad tienen que ver con la conectividad de las reglas, por lo tanto, involucran las cadenas de inferencia y debe tomarse en consideración, durante su diseño, la estrategia de encadenamiento definida por el lenguaje. Como se muestra en el cuadro, VBCH implementa encadenamiento hacia atrás (EA), como casi todas las herramientas analizadas, excepto EVA y KB-REDUCER que poseen encadenamiento hacia delante (ED). En el caso de IMVER hay manejo de ambas estrategias.

VBCH detecta *anomalías* que tienen que ver con redundancia (RE), circularidad (CI) e inconsistencia (IC) en la base, aunque también incorpora el examen de valores ilegales. Como problemas de redundancia localiza: reglas duplicadas e incluidas, y de inconsistencia: contradicciones y condiciones innecesarias (este análisis sólo se realiza en HESV).

La detección de ciclos es otra de las diferencias que presenta VBCH con el resto de las herramientas, en cuanto a la cantidad y el tipo de estructuras de representación involucradas en dicha anomalía. Muchas de las herramientas analizadas efectúan la detección de circularidad para pares de reglas (como es el caso de VALENS), es decir, no son capaces de detectar ciclos formados por más de dos reglas. Además el lenguaje HAries define otro tipo de estructuras denominadas proposiciones. Éstas permiten incorporar relaciones que aunadas a las reglas en el proceso de inferencia pueden causar circularidad. Las definiciones de las estructuras en HAries se presentan en el siguiente capítulo.

En cuanto a la *implementación* casi todas las herramientas han sido construidas con Prolog, C o la combinación de éstos, excepto KB-REDUCER en el que se ha empleado como lenguaje Lisp y VALENS que utiliza Aion. De CRIB y HESV no se ha encontrado esta información en la documentación disponible, lo cual se indica en el cuadro con el símbolo: ?

[1]La forma del antecedente se profundiza en 3.2.2

2.4. Conclusiones

Luego del estudio realizado en este capítulo, se han elegido los métodos de verificación más apropiados, que han sido desarrollados en función de las características de los esquemas de representación que posee el lenguaje HAries. Por otra parte, se examinaron las soluciones que brindan las herramientas a determinados problemas que se presentan en una BC.

Además el análisis permitió proporcionar un panorama general de los trabajos relacionados y ubicar al sistema VBCH dentro de un conjunto de herramientas de verificación.

Capítulo 3

Lenguaje de Representación del Conocimiento: HAries

HAries es un lenguaje de representación del conocimiento, que posibilita el desarrollo de sistemas inteligentes en diversos dominios del conocimiento [1], [24]. Consta de varias herramientas en aspectos tales como la representación de conocimientos, las estrategias de control, el razonamiento con incertidumbre, el aprendizaje automático y otras [25].

El ingeniero del conocimiento dispone así de un conjunto amplio y variado de medios que, de forma individual o combinada, pueden ser empleados bien en la construcción de sistemas inteligentes concretos en diferentes campos, o bien en la experimentación, el adiestramiento o la propia investigación de esta disciplina.

En este capítulo se presentan las características del lenguaje y las estructuras de representación del conocimiento que utiliza. Una vez introducidas las estructuras, se describen los componentes que las conforman, la sintaxis y la semántica de las mismas. Por último, se explica a grandes rasgos la forma en que se lleva a cabo el manejo de incertidumbre y las características básicas del proceso evaluativo.

3.1. Características generales de HAries

El concepto fundamental que se ha utilizado en el desarrollo de HAries, parte de suponer, que toda fuente de información extraída de un problema, puede ser considerada como conocimiento en algún sentido. De esta forma, se pueden utilizar datos, experiencias humanas y diversos medios computacionales como: imágenes, sonidos y animaciones de forma híbrida [1], [26]. Si a esto se añaden sus posibilidades de incluir conocimiento obtenido por la vía del aprendizaje automático (más específicamente, inductivo), se hace evidente que el espectro de conocimientos que puede construirse en este sentido es muy variado.

3.1.1. Componentes principales

Atendiendo a su concepción general, los principales componentes de HAries son:

- Sistema de control general (HAries)
- Sistema de adquisición y manipulación de conocimientos (HAriesA)
- Sistema consultante para el desarrollador (HAriesC)
- Utilitarios

El *sistema de control general*, constituye la primera vía disponible para entrar al sistema. A través de éste se brinda acceso al resto de los componentes u operaciones que proporciona el medio ambiente. Este módulo también incluye algunas operaciones específicas como son: instalación de las aplicaciones y un traductor. Este último permite convertir versiones anteriores hacia la actual de forma automática.

El *sistema de adquisición y manipulación de conocimientos* permite ejecutar todas las operaciones relativas a la creación, modificación y mantenimiento de las BC. Incluye varios editores visuales, compiladores y decompiladores, que facilitan el proceso de programación, evitando que el usuario tenga que trabajar con el código fuente directamente.

El *sistema consultante para el desarrollador* constituye la parte del ambiente que se encarga del procesamiento de la información almacenada en las BC. Contiene diversos mecanismos de inferencia y razonamiento con incertidumbre, además de un complejo proceso de control a diversos niveles jerárquicos.

Los *utilitarios* forman un conjunto de funciones o módulos, que sirven como apoyo al proceso de creación y mantenimiento de las aplicaciones, los fundamentales son: sistemas explicatorios y agentes buscadores de información. El sistema desarrollado, VBCH, formará parte de los utilitarios del lenguaje.

3.1.2. Estructuras de representación del conocimiento

HAries basa su funcionamiento en una gran diversidad de estructuras, que permiten la representación de conocimientos del mundo real en la computadora [27], [28], siendo la BC el ente computacional, donde se almacenan las informaciones necesarias sobre el problema en cuestión, en correspondencia con las estructuras de representación que define el lenguaje.

El término Base de Conocimientos en HAries se interpreta como la unión de dos componentes fundamentales:

1. Un conjunto formado por *hechos* o *conceptos* provenientes de un dominio del conocimiento específico y

2. Un conjunto de *relaciones* entre los elementos del conjunto anterior.

En las próximas secciones se describen las dos estructuras principales del lenguaje: proposición y regla de producción, las cuales expresan los elementos de los conjuntos de hechos y relaciones respectivamente.

Es importante destacar que HAries no proporciona símbolos de funciones ni de variables que permitan representar acepciones generales, característica propia del cálculo de predicados.

3.1.3. Razonamiento con incertidumbre

Uno de los aspectos más importantes en la problemática de los SBC es el de la representación y procesamiento de la incertidumbre, lo cual se lleva a cabo en HAries empleado una teoría algebraica de funciones de contribución[1] [25] mezclada con aspectos de lógica difusa.

En términos de significado de los hechos que se manejan en la BC, esto equivale a establecer una gradación continua del valor de verdad desde -1 (totalmente falso) a 1 (totalmente cierto).

El procesamiento de la incertidumbre en el lenguaje se describe en las secciones 3.2.3 y 3.4 de este capítulo.

3.2. Proposición

Las primeras unidades de conocimiento a construir cuando se desarrolla un sistema inteligente, son las estructuras simples o irreducibles que definen el problema a resolver.

Una *proposición simple* no es más que un planteamiento o aseveración que expresa nuestro criterio, opinión, juicio o descripción sobre el objeto de estudio que puede, en un momento determinado, resultar verdadero o falso. [25].

Una proposición se caracteriza por:

1. Representar un concepto de la realidad.

2. Tener asociado un valor de certidumbre, que exprese el grado de veracidad con que se cumple la situación planteada. Este valor se representa como $C(P)$ y pertenece al intervalo $[-1, 1]$, siendo P la proposición en cuestión.

[1] Desarrollada por el matemático checo P. Hajek en 1985

3.2.1. Elementos de la proposición

Los elementos que conforman la estructura proposición son:

$$P =< T, C, AT, RL, EA, AC >$$

- Texto (T): Constituyen la definición en lenguaje natural de lo que se está representando.
- Certidumbre (C): Es valor de verdad asociado a una proposición, con el cual se podrá identificar si un hecho determinado, que se encuentre representado según esta estructura, se satisface ($C(P) > 0$) o no ($C(P) \leq 0$).
- Atributos (AT): Representan determinadas propiedades que posee la proposición. El primer atributo (natural) está en correspondencia con las relaciones que se establecen con otras proposiciones a través de las reglas. De acuerdo a esto, una proposición puede ser [25]:
  - *Pregunta*: La proposición sólo aparece en el antecedente de al menos una regla. Las proposiciones preguntas constituyen el conjunto de todas las posibles entradas de la BC, cuyos valores son proporcionados por el usuario del sistema.
  - *Objetivo*: Sólo aparece en el sucedente de al menos una regla, pero no ocurre en el antecedente de ninguna otra. Las proposiciones objetivos definen el conjunto de todas las salidas de la BC (pueden ser interpretadas como el conjunto de conclusiones).
  - *Intermedio*: Aparece tanto en el antecedente como en el sucedente de alguna regla (no la misma). Las proposiciones intermedios necesitan ser inferidas para evaluar los objetivos y utilizan los valores de las preguntas u otros intermedios para ello.

  El segundo atributo (dinamismo) tiene que ver con la monotonía en la evaluación de la proposición. Conforme a esto, se puede clasificar a la proposición como:
  - *Estática*: Si el valor de certidumbre de la proposición es obtenida una sola vez durante el proceso de solución de un caso dado o
  - *Dinámica*: Si cada vez que se necesite su valor éste tiene que ser calculado u obtenido de nuevo, lo cual implica razonamiento no monótono.
- Relaciones con otras proposiciones (RL): Permiten la representación de situaciones que no pueden ser tratadas con reglas de producción. Existen dos tipos fundamentales:
  - *Contextuales*: Establecen las condiciones bajo las cuales puede variar el proceso de evaluación natural[1] de una proposición.

[1]Proceso evaluativo que emplea el sistema siempre que no existan relaciones y se explica a detalle en la sección 3.4

Una relación contextual respecto a una proposición P es la expresión $P(W)@c$, y significa que: "se evalúa P siguiendo el proceso natural de evaluación si y sólo si se cumple la condición de análisis expresada en c ($C(c) > 0$) y en caso contrario, P se evalúa tomando el valor de certidumbre alterno W".

Ejemplo: Tomando la relación existente entre las proposiciones siguientes:

1. La paciente está embarazada
2. El paciente es un hombre

Se observa que la proposición uno está en el contexto de la negación de dos, lo que en términos del lenguaje se expresa como: $1(-1)@-2$, lo cual significa que el análisis del embarazo de la paciente (proposición 1), se desencadena sólo si se conoce que es una mujer (negación de proposición 2), puesto que en caso contrario, se puede asegurar con completa seguridad que no está embarazada ($C(1) = -1$).

- *Evaluación alternativa*: Afecta únicamente la forma de evaluar P en correspondencia con una condición.

Una relación de evaluación alternativa con respecto a una proposición P es la expresión $P\xi c$, y significa que: "si se cumple c, P se evalúa como pregunta directa al usuario, independientemente de si existen o no relaciones con otras proposiciones que posibiliten su deducción; mientras que de no cumplirse la condición dada, entonces no se modifica el modo de evaluación natural".

Ejemplo: Considerando un problema de diagnóstico de epilepsia, es posible que en algunos casos el usuario del sistema (especialista) tenga algún criterio formado y en otros no, en cuyo caso sería el SBC quien acometería la tarea de determinar el diagnóstico del paciente. Entonces para las siguientes proposiciones:

1. El paciente tiene epilepsia
2. Ud. tiene un diagnóstico seguro

Se podría definir entonces esta situación como: $1\xi 2$, lo que significa que de existir un diagnóstico seguro sobre la salud del paciente (proposición 2) entonces se le puede preguntar al usuario lo que sabe con relación a este hecho (proposición 1), pero en caso de no existir esta seguridad, ello sea evaluado por el SBC utilizando el conocimiento que posea sobre el particular.

- Elementos asociados (EA): Una proposición puede establecer relaciones con otras proposiciones, o incluso elementos de estructuras diferentes. A una proposición se le puede asociar:

  - La primera regla de producción donde la proposición aparece en el sucedente, que es el punto de partida para la evaluación de la proposición.

  - Otras proposiciones asociadas, que deben ser evaluadas antes de iniciar el proceso evaluativo de la proposición como tal.

- Conjuntos de acciones (AC): Son acciones que deben ser ejecutadas antes de iniciar el proceso evaluativo de la proposición en cuestión y/o después de concluir éste.

3.2.2. Sintaxis de las proposiciones

La sintaxis permite declarar, categorizar y sugerir una primera interpretación de los datos en forma abstracta. A continuación se describe la sintaxis de cada uno de los elementos que conforman la estructura proposición.

- Texto $T ::= ASCII^*$
- Certidumbre $C ::= [-1, 1] \in R$
- Atributos $AT ::=$ {Natural: Pregunta | Intermedio | Objetivo, Dinamismo: Estática | Dinámica}
- Condiciones $c ::= p \qquad p \in Proposiciones$
  $\mid -p$
  $\mid c_1 \,\&\, c_2 \qquad c_1, c_2 \in c$
  $\mid c_1 \, V \, c_2$
  $\mid c_1 \mid c_2$
- Relaciones $RL ::= p(w) \,@\, c \qquad p \in Proposiciones$
  $\mid p \, \xi \, c$
- Elementos asociados $EA ::= p^* r \qquad p \in Proposiciones, \quad r \in Reglas$
- Acciones $AC ::=$ {Mensaje | Hipertexto | Imagen | Video | Sonido | Programa externo}
- Proposiciones simples $P ::= < T, C, AT, RL, EA, AC >$

A partir de las proposiciones simples es posible construir nuevas proposiciones compuestas, introduciendo diversos conectivos como: $-$ para la negación, & para la conjunción(se satisface si se cumplen todas las proposiciones), V para la disyunción inclusiva (se satisface si se cumple al menos una proposición) y $\mid$ para la disyunción excluyente (se satisface si se cumple sólo una proposición).

El caso más simple de proposición compuesta se obtiene con la definición de literal. Un *literal* L es una proposición simple P o su negación $-P$. Analicemos ahora cómo se pueden construir proposiciones compuestas utilizando el resto de los conectivos.

- *Conjunción elemental* (CE): Es una expresión de la forma $L_1 \,\&\, L_2 \,\&\, ... \,\&\, L_n$
- *Disyunción inclusiva elemental* (DIE): Es la expresión $L_1 \, V \, L_2 \, V \, ... \, V \, L_n$
- *Disyunción excluyente elemental* (DEE): Se expresa como $L_1 \mid L_2 \mid ... \mid L_n$

Además, se pueden introducir proposiciones compuestas mucho más complejas, utilizando de forma combinada las anteriores. Las siguientes expresiones también se admiten como válidas en el lenguaje y se consideran proposiciones compuestas:

- *Forma conjuntiva elemental* (FCE): $\overset{s}{\&} K_i$ donde K_i puede ser:
  - Un literal L
  - Una disyunción inclusiva elemental DIE
  - Una disyunción excluyente elemental DEE
- *Forma disyuntiva inclusiva elemental* ($FDIE$): $\overset{s}{V} K_i$ donde K_i puede ser
  - Un literal L
  - Una conjunción elemental CE
  - Una disyunción excluyente elemental DEE
- *Forma disyuntiva excluyente elemental* ($FDEE$): $\overset{s}{|} K_i$ donde K_i puede ser
  - Un literal L
  - Una conjunción elemental CE
  - Una disyunción inclusiva elemental DIE

En todos los casos "s" representa la cantidad de partes o expresiones K_i que integran las formas.

3.2.3. Semántica de las proposiciones

El conjunto de significados con que se valora el contenido de información de una proposición simple está dado por el valor de certidumbre asociado a dicha proposición. Este valor es un número del intervalo [-1,1], donde 1 indica verdad absoluta, -1 falsedad absoluta y 0 desconocimiento total. El resto de los valores se encuentran asociados a escalas de veracidad, que corresponden a la afirmación (valores positivos) o negación (valores negativos) según el caso.

De acuerdo con las consideraciones anteriores, se define a continuación la forma en que los distintos conectivos generan los significados de las proposiciones compuestas a partir de las simples que conectan.

- Función de la *negación* del peso de una proposición $P : NEG(C(P))$
  donde: $NEG(C(P)) = -C(P)$
- Función de la *conjunción* de los pesos de literales $L_1, ..., L_k : CONJ(C(L_1), ..., C(L_k))$
  donde: $CONJ(C(L_1), ..., C(L_k)) = MIN\{C(L_1), ..., C(L_k)\}$
- Función de la *disyunción inclusiva* de los pesos de $L_1, ..., L_k : DISYI(C(L_1), ..., C(L_k))$
  donde: $DISYI(C(L_1), ..., C(L_k)) = MAX\{C(L_1), ..., C(L_k)\}$
- Función de la *disyunción excluyente* de los pesos de $L_1, ..., L_k : DISYE(C(L_1), ..., C(L_k))$
  donde: $DISYE(C(L_1), ..., C(L_k)) = MIN\{|C(L_1)|, ..., |C(L_k)|\}$

Para las formas combinadas de los conectivos, se calcula la certidumbre asociada a cada parte (K_i) y luego se aplican las mismas funciones para el conectivo principal.

3.3. Regla de producción generalizada

La regla de producción es la forma más popular de representación del conocimiento utilizada en los SBC y ello es debido a la uniformidad de su estructura y la naturalidad con que se expresan conocimientos principalmente procedimentales.

"Una representación procedimental es aquella en la que la información de control necesaria para utilizar el conocimiento se encuentra embebida en el propio conocimiento" [5].

Las reglas de producción constituyen sentencias del tipo "si condición entonces acción" y el término *generalizada* se debe a que esta estructura tiene la característica de poseer un volumen de información, que clásicamente en otros sistemas, reclamaría la definición de varias reglas para representar la misma situación.

3.3.1. Elementos de la regla de producción generalizada

Se llama *Regla de Producción Generalizada* (RPG) a toda relación de la forma:

$$A \Rightarrow S_1(WS_{11}\ WS_{12}), ..., S_r(WS_{r1}\ WS_{r2})$$

donde: A es el antecedente, $S_1, ..., S_r$ los sucedentes y $WS_{ij} \in [-1, 1](i = 1, ..., r; j = 1, 2)$, son los valores de certidumbre (también llamados pesos) que cuantifican la relación de producción establecida por el símbolo $\Rightarrow$ entre el antecedente y el sucedente [25].

La estructura del antecedente puede contener una proposición o composición de proposiciones mediante conectivos lógicos y el sucedente una o varias proposiciones simples.

En relación a los valores de certidumbre, el primer peso (W_{i1}) indica la contribución al grado de certidumbre total de la proposición sucedente i si la premisa del antecedente es absolutamente cierta; mientras que el segundo peso (W_{i2}) indica la contribución al sucedente i si el antecedente no se cumple.

3.3.2. Sintaxis de las reglas de producción generalizadas

- Literales $L ::= p \qquad p \in Proposiciones\ simples$
  $\qquad | - p$
- Conjunción elemental $CE ::= l_1 \,\&\, l_2 \qquad l_1, l_2 \in L$
  $\qquad |ce_1 \,\&\, ce_2 \qquad ce_1, ce_2 \in CE$
- Disyunción inclusiva elemental $DIE ::= l_1\ V\ l_2 \qquad l_1, l_2 \in L$
  $\qquad |die_1\ V\ die_2 \qquad die_1, die_2 \in DIE$
- Disyunción excluyente elemental $DEE ::= l_1 \mid l_2 \qquad l_1, l_2 \in L$
  $\qquad |dee_1 \mid dee_2 \qquad dee_1, dee_2 \in DEE$
- Forma conjuntiva elemental $FCE ::= \overset{s}{\&}\ K_i \qquad K_i \in L, DIE, DEE$
- Forma disyuntiva inclusiva elemental $FDIE ::= \overset{s}{V}\ K_i \qquad K_i \in L, CE, DEE$
- Forma disyuntiva excluyente elemental $FDEE ::= \overset{s}{|}\ K_i \qquad K_i \in L, CE, DIE$
- Antecedentes $A ::= l \qquad l \in L$

$$\begin{array}{ll} |ce & ce \in CE \\ |die & die \in DIE \\ |dee & dee \in DEE \\ |fce & fce \in FCE \\ |fdie & fdie \in FDIE \\ |fdee & fdee \in FDEE \end{array}$$

- Sucedentes $S ::= p^{+}(W_{i1}\ WS_{i2})$
- Reglas de Producción Generalizadas $RPG ::= A \Rightarrow S$

3.3.3. Semántica de las reglas de producción generalizadas

Una RPG $A \Rightarrow S_1(WS_{11}\ WS_{12}), ..., S_r(WS_{r1}\ WS_{r2})$ expresa lo siguiente: "Si se está absolutamente seguro de que se cumple el antecedente A entonces se puede afirmar que esto contribuye en WS_{11} al grado de certidumbre global de S_1, en WS_{21} al grado de certidumbre global de S_2, y así sucesivamente para los r sucedentes, y si por el contrario se está absolutamente seguro de que No se satisface A entonces eso contribuye en WS_{12} al peso de S_1, en WS_{22} al peso de S_2 y de la misma forma hasta S_r [26]."

Es importante destacar, que los pesos WS no representan los valores de certidumbre asociados a las proposiciones sucedentes en caso de cumplirse el antecedente con un valor extremo, sino la contribución que se obtiene en tal situación. Como una proposición puede pertenecer al sucedente de varias reglas y cada una aportará su certidumbre, el valor final de ésta se obtendrá como producto de la combinación de todos estos aportes. En otras palabras, para obtener una conclusión, se deben analizar todas las evidencias relacionadas con ella y encontrar un resultado unificado.

3.4. Proceso evaluativo

Las proposiciones son estructuras pasivas en cuanto a que por sí mismas no son capaces de obtener su valor de certidumbre, esto se realiza a partir de las relaciones que se establecen entre las reglas de producción.

Considerando que las reglas de producción constituyen una representación procedimental se requiere de un intérprete que siga las instrucciones dadas por el conocimiento. Este intérprete puede ser construido empleando como esquema de control el encadenamiento hacia atrás o hacia delante [5].

En HAries se implementa encadenamiento hacia atrás, enfocando la atención en el primer objetivo de la BC y considerando la secuencia de reglas en las que la proposición objetivo forme parte del sucedente para su evaluación.

El encadenamiento hacia atrás define un camino de búsqueda concreto denominado *cadena de inferencia* para así encontrar una respuesta a cualquier pregunta. Si para un determinado objetivo no hay reglas que permitan su deducción, el sistema consulta al usuario para su evaluación. En caso contrario se realiza el análisis de cada regla.

Ahora bien, el análisis de una regla implica el de su antecedente y en particular, el de las proposiciones que lo componen, planteándose dos posibilidades: que la proposición sea una pregunta, en cuyo caso su evaluación se resuelve mediante la interrogación directa al usuario, o que sea un intermedio, lo cual provoca la definición de un subobjetivo a ser analizado, que se evalúa aplicando el mismo proceso recursivamente [25]. Una vez obtenido el peso de las proposiciones necesarias se calcula el correspondiente

al antecedente, usando las funciones NEG, CONJ, DISYI y DISYE o su combinación según el caso.

El segundo aspecto a considerar, viene dado entonces, por la situación en la cual se tiene una certidumbre no absoluta en el antecedente. De lo anterior se desprende que se necesita definir una función para calcular la contribución que proporciona una regla al peso del sucedente, en correspondencia con el del antecedente. Dicha función se denomina función de contribución (se denota CTR).

Para completar la evaluación se necesita analizar cómo calcular el peso final de la proposición objetivo, conocidas previamente todas las contribuciones de las diferentes reglas que conducen a ellas. Ahora bien, si existe una sola, la CTR define el peso, pero si no, se necesita una función de contribución global (GLOB) para combinar todos los aportes y obtener un resultado único.

Debido a que la representación y procesamiento de la incertidumbre, no constituye el objetivo del trabajo, sino un complemento que utilizamos para ganar unidad en el mismo, no se desarrollan aquí estos aspectos en detalles, que se encuentran recogidos en los apéndices A y B.

Hasta el momento se ha explicado la forma en que se analizan las reglas partiendo de un objetivo, pero no se ha mencionado el orden en que se ejecutan dichas reglas (en caso de que el objetivo en cuestión aparezca en el sucedente de más de una), para ello en HAries se utiliza como método de resolución de conflictos la forma más simple de asignar preferencias basándose en las propias reglas, que consiste en dar prioridad a las reglas según el orden en el que aparecen almacenadas dentro de la BC.

Todo este análisis describe la forma en que se resuelve una consulta a un SBC en HAries, cuyo problema consiste en inferir el valor de certidumbre asociado a cada una de las proposiciones objetivo que formen parte de dicha consulta, a lo cual se le denomina *proceso evaluativo* y constituye el resultado que brindará el SBC.

Ejemplo: Veamos un ejemplo utilizando una representación por medio de grafos de una base de reglas hipotética que se muestra en la figura 3.1. Ésta consta de 17 proposiciones y las 11 reglas que se muestran a continuación:

$$
\begin{aligned}
&R1: \ 15 \mid 17 \Rightarrow 11(-1\ 1);\\
&R2: \ 8 \Rightarrow 11(0.9\ -0.3);\\
&R3: \ (1\ V\ 12)\ \&\ (12\ V\ 4) \Rightarrow 16(1\ 0), 8(-1\ 0);\\
&R4: \ 1\ \&\ 9 \Rightarrow 6(0.7\ 0);\\
&R5: \ 3\ V\ 5 \Rightarrow 2(-0.9\ 0.7);
\end{aligned}
$$

$R6:\ 11 \Rightarrow 7(1\ 0.8);$
$R7:\ -4\ \&\ 16 \Rightarrow 15(1\ -1);$
$R8:\ 6\ V\ -1 \Rightarrow 10(1\ 0);$
$R9:\ 5\ V\ 12 \Rightarrow 7(0.9\ -1), 2(-0.8\ 0);$
$R10:\ 10 \Rightarrow 3(0.8\ -0.8);$
$R11:\ 14 \Rightarrow 6(1\ -1);$

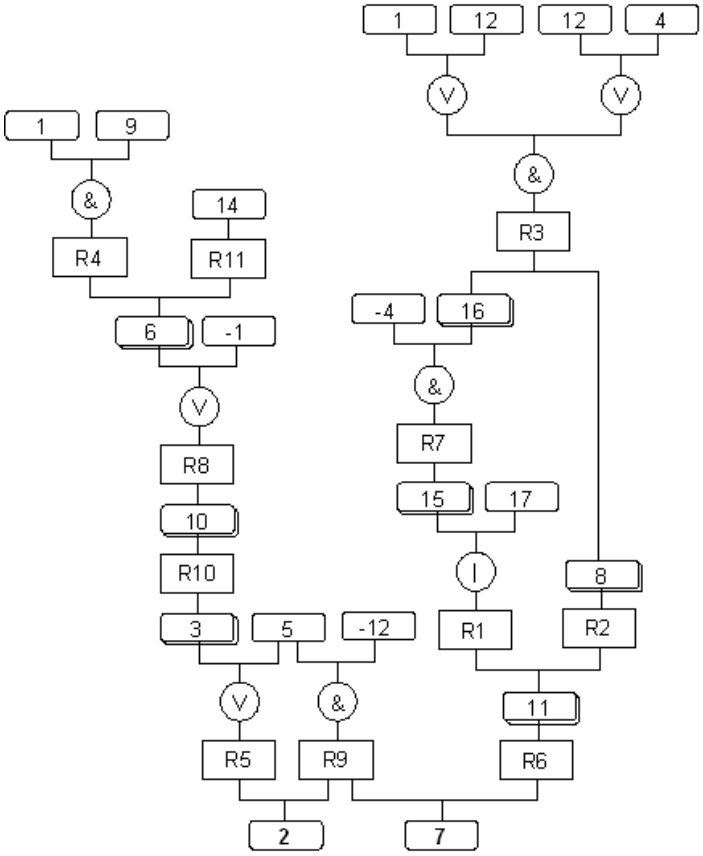

Figura 3.1: Representación gráfica de una BC hipotética.

Nota: Las proposiciones que se encuentran con doble rectángulo son proposiciones intermedios. El resto son proposiciones preguntas, excepto 2 y 7 son objetivos.

El cuadro 3.1 muestra cómo funcionaría el mecanismo descrito para evaluar la proposición objetivo 2. En el cuadro pueden observarse uno por uno los pasos y resultados en cada caso.

Pasos	Acción				Resultados	
	Mandar evaluación de		Ejecutar	Posponer	Concluida evaluación de	
	Regla	Proposición			Regla	Proposición
1		**2**		×		
2	5			×		
3		3		×		
4	10			×		
5		10		×		
6	8			×		
7		6		×		
8	4			×		
9		1(p)	×			1
10		9(p)	×		4	9
11	11			×		
12		14(p)	×		11	14, 6
13		1	×		8, 10	10, 3
14		5(p)	×		5	5
15	9			×		
16		5	×			
17		12(p)	×		9	12, **2**

Cuadro 3.1: Proceso evaluativo para una proposición objetivo

En el primer paso se plantea la evaluación del objetivo 2, acción que debe posponerse pues no se cuenta con los valores de certidumbre de las proposiciones que conducen al mismo. Se ejecuta entonces la evaluación de la regla 5 que es la primera en la cual 2 se encuentra como parte del sucedente. La regla 5 a su vez requiere de los valores de sus proposiciones antecedentes: 3 y 5, por tanto se comienza tratando de evaluar a 3 que es una proposición intermedio y se ejecuta para ello el mismo procedimiento que hasta el momento para determinar 2. Los puntos de detección de este mecanismo se encuentran cuando hay que evaluar una proposición pregunta, denotadas en la tabla por (p). Es precisamente en estos puntos donde el proceso encuentra nodos de fin y se comienza a retornar para el reanálisis de las tareas pospuestas.

La figura 3.2 es un resumen del manejo de incertidumbre en HAries durante el proceso evaluativo. Las incompatibilidades entre reglas que involucran análisis de incertidumbre también se incluyen en este esquema y serán explicadas con detalle en el próximo capítulo.

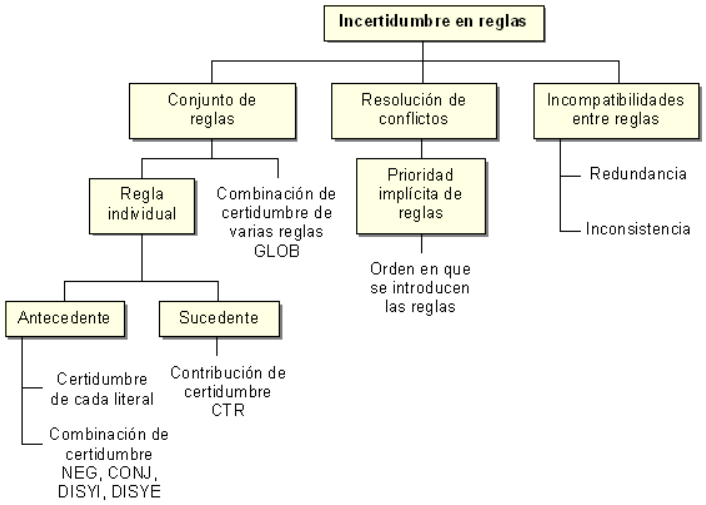

Figura 3.2: Incertidumbre en las reglas.

3.5. Conclusiones

HAries está concebido como un medio ambiente de propósito general y como un lenguaje para la programación del conocimiento enfocado hacia la construcción de SBC concretos y en cierto sentido también, hacia la enseñanza de los principios, ideas y arquitecturas de los mismos.

En este capítulo se presentaron las características generales del lenguaje y algunos aspectos de sus dos estructuras de representación del conocimiento más importantes: proposiciones y reglas de producción generalizadas. Principalmente se hizo el análisis con relación a la sintaxis, la semántica y el proceso evaluativo durante una consulta.

Capítulo 4

Proceso de Verificación

Para la definición acertada de los tipos y subtipos de anomalías que pueden presentarse en una BC, descrita con el lenguaje HAries, es importante comenzar revisando los principios de la verificación.

Las suposiciones comúnmente aceptadas sobre la verificación de una BC son [30]:

1. Tanto la sintaxis como la semántica de las anomalías deben ser definidas en términos de la sintaxis y la semántica del lenguaje de representación del conocimiento usado para expresar la BC.

2. Las anomalías son definidas en términos del significado declarativo de la BC, en vez de cualquier significado procedural.

3. Las anomalías son detectadas examinando la sintaxis de la BC, aunque éstas deben ser entendidas semánticamente.

4. Los métodos de detección de anomalías se aplican solamente a la BC del SBC, aunque se deben tomar en cuenta ciertas propiedades de la máquina de inferencia.

5. No todas las anomalías son errores: son síntomas de probables errores en la BC.

Una vez revisados estos fundamentos, es necesario definir todos los elementos involucrados en las anomalías, en términos de la sintaxis de la BC especificada por el lenguaje HAries.

4.1. Conceptos previos

Definición 1 *Una* Base de Conocimientos *en HAries está definida como la unión de una base de una base de reglas $\mathcal{R}$ y proposiciones $\mathcal{P}$.*

Definición 2 *Una* base de reglas $\mathcal{R}$ *es un conjunto de expresiones* $R_1, R_2, ..., R_n$ *llamadas reglas de producción generalizadas y definidas anteriormente*[1] *de la forma:*

$$A \Rightarrow S_1(WS_{11}WS_{12}), ..., S_n(WS_{n1}WS_{n2}).$$

En una regla se definen tres funciones:

$$\begin{aligned} ant(R) &= \{P_1, P_2, ..., P_n\} \\ suc(R) &= \{P_1, P_2, ..., P_n\} \\ cert(suc(R)_i) &= WS_{i1}, WS_{i2} \end{aligned}$$

Donde:

- $P_1, P_2, ..., P_n$ son las proposiciones que forman parte del antecedente o sucedente según el caso y
- $suc(R)_i$ hace referencia a un elemento del conjunto definido en $suc(R)$.

Definición 3 *Una* base de proposiciones *es un conjunto de expresiones* $\mathcal{P} = \mathcal{O} \cup \mathcal{Q} \cup \mathcal{I}$, *donde* $\mathcal{O}$ *es el conjunto de proposiciones objetivos,* $\mathcal{Q}$ *el conjunto de proposiciones preguntas e* $\mathcal{I}$ *el conjunto de proposiciones intermedios*[2].

Definición 4 *El conjunto de* restricciones semánticas $\mathcal{C}$ *está formado por expresiones* $C_1, C_2, ..., C_n$ *llamadas relaciones contextuales y de evaluación alternativa, definidas anteriormente*[1] *de la forma:*

$$\begin{aligned} &P(W) \ @ \ c \\ &P \ \xi \ c \end{aligned}$$

En estas relaciones se definen dos funciones:

$$\begin{aligned} asoc(C) &= P \\ cond(C) &= \{P_1, P_2, ..., P_n\} \end{aligned}$$

Donde:

- P indica la proposición a la cual está asociada la relación y
- $P_1, P_2, ..., P_n$ las proposiciones que forman parte de la condición c en la relación.

[1] Véase sección 3.2.1
[2] Esta clasificación fue explicada en 3.2.1

Definición 5 *Un* ambiente $\mathcal{E}$ *es el subconjunto de las literales cuyos valores son conocidos en un momento dado y que no implican contradicciones semánticas.*

Definición 6 *El valor de una proposición* $P \in \mathcal{P}$ *que posea una restricción semántica* $C \in \mathcal{C}$ *sólo se puede* inferir *si existe algún ambiente* $E \in \mathcal{E}$ *tal que la evaluación de la condición de* C *es una consecuencia lógica de proporcionar* E *como entrada de* $\mathcal{C}$*. Formalmente,*

$$\exists P \in \mathcal{P}, C \in \mathcal{C}, E \in \mathcal{E}$$
$$(asoc(C) = P \wedge infiere(P, \mathcal{C}, E) ssi (\mathcal{C} \cup E) \vdash cond(C))$$

Definición 7 *El sucedente de una regla* $R \in \mathcal{R}$ *sólo se puede* inferir *si existe algún ambiente* $E \in \mathcal{E}$ *tal que la evaluación del antecedente de* R *es una consecuencia lógica de proporcionar* E *como entrada de* $\mathcal{R}$*. Formalmente,*

$$\exists P \in \mathcal{P}, R \in \mathcal{R}, E \in \mathcal{E}$$
$$(P \subset suc(R) \wedge infiere(P, \mathcal{R}, E)\ ssi\ (\mathcal{R} \cup E) \vdash ant(R))$$

4.2. Clasificación y definición de anomalías

Las anomalías que se pueden presentar en las estructuras de representación estudiadas y sus relaciones, han sido clasificadas en tres categorías, como se muestra en la figura 4.1.

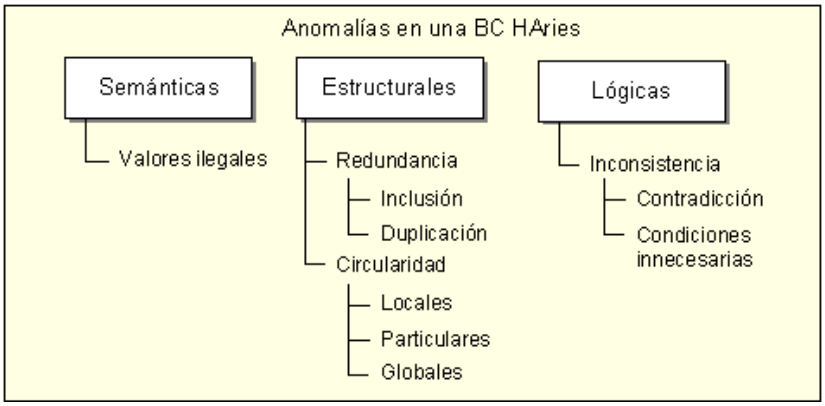

Figura 4.1: Clasificación de anomalías en HAries.

- Dentro de los errores semánticos se encuentran los *valores ilegales* que no son más que referencias a estructuras que no han sido declaradas en la base.

- En las anomalías estructurales se hallan la *redundancia* y *circularidad*.

Los tipos de redundancia que detecta la herramienta de verificación son: duplicación e inclusión, que son exclusivas de la representación por reglas de producción. En cuanto a los ciclos se identificaron en HAries tres categorías: locales, particulares y globales. Los ciclos locales involucran un único elemento de la estructura, por ejemplo una regla específica. Los ciclos particulares involucran varios elementos de una misma estructura y finalmente los globales involucran elementos de estructuras diferentes, por ejemplo reglas y relaciones en las proposiciones.

- Los problemas lógicos están relacionados con inconsistencias en la BC como contradicciones y condiciones innecesarias en la reglas.

Dos reglas son contradictorias cuando para la misma condición las acciones presentan contribuciones diferentes a los grados de certidumbre de las proposiciones sucedentes. Las condiciones innecesarias representan proposiciones que pueden ser eliminadas del antecedente, sin causar modificación alguna en el proceso de inferencia.

4.3. Valores ilegales

Los rangos ilegales se pueden encontrar en las dos estructuras de representación y en el establecimiento de relaciones entre ellas. Tomando por ejemplo, el análisis de las proposiciones en la base de reglas tenemos:

Definición 8 *En una base de reglas $\mathcal{R}$ encontramos* valores ilegales *o fuera de rango si para alguna regla $R \in \mathcal{R}$ se localiza un literal L que no pertenece a la base de proposiciones $\mathcal{P}$:*

$$\exists L, R \in \mathcal{R}((L \subset ant(R) \vee L \subset suc(R)) \wedge L \notin \mathcal{P})$$

Ejemplo: En una base de conocimientos con 10 proposiciones, la siguiente regla contendrá un valor ilegal para la proposición identificada con el número 12, puesto que el identificador de las proposiciones se corresponde con la posición que ocupan dichas proposiciones en la BC.

$$R3: \ 4 \ \& \ 12 \ \& \ (2 \ V \ 10) \Rightarrow 8(-1 \ 1)$$

Este mismo análisis se hace para las condiciones de las restricciones.

Definición 9 *En una base de proposiciones $\mathcal{P}$ encontramos* valores ilegales *o fuera de rango si en la condición de alguna restricción $C \in \mathcal{C}$ se localiza un literal L que no pertenece a la base de proposiciones $\mathcal{P}$:*

$$\exists L, C \in \mathcal{C}(L \subset cond(C) \wedge L \notin \mathcal{P})$$

4.3.1. Métodos de detección de valores ilegales

En el caso de la localización de los valores ilegales en la base de reglas, se realiza un recorrido total o parcial según se indique en el intervalo de verificación (*IniR* - *FinR*) y se efectúa el análisis de los literales como se muestra en el algoritmo 1. Durante el análisis de cada regla se va examinando que los literales que forman parte del antecedente *Ant* y sucedente *Suc* hayan sido definidas en la base de proposiciones (sean menores que $NProp$).

Algoritmo 1 Comparación de literales en una regla con el número de proposiciones de la BC

Entrada: La regla que se analiza, R; el número de proposiciones de la base, $NProp$.
Salida: Valor booleano que indica si se encontraron proposiciones fuera de rango, la lista de estas proposiciones $ListaErr$.

ComparaLiterales($R, NProp, \&ListaErr$)
$Ant \leftarrow$ Extrae_antecedente(R)
$Suc \leftarrow$ Extrae_sucedente(R)
for cada proposición p en el antecedente Ant **do**
  if $p > NProp$ **then** {Se encontró una proposición fuera de rango}
    $ListaErr \leftarrow$ Add(P)
for cada proposición p en el sucedente Suc **do**
  if $p > NProp$ **then**
    $ListaErr \leftarrow$ Add(P)
if NOT $ListaErr$ **then** {No se encontró error}
  return FALSE
else
  return TRUE

Para la detección de valores ilegales en la base de proposiciones, se recorre la lista de éstas estructuras y se realiza el análisis de todas las relaciones que pueden ser definidas por el lenguaje, como se presenta en el algoritmo 2. En ComparaLiterales se va examinando que en la lista de proposiciones asociadas *PropAsoc*, la relación contextual *Contex* y la de evaluación alternativa *EvalAlter* de P (en caso de que éstas hayan sido definidas), no existan literales mayores que *NProp*.

4.4. Redundancia

Definición 10 *Dadas dos reglas $R, R' \in \mathcal{R}$, se dice que R está* incluida *en R' si su antecedente está contenido en el antecedente de R' y todos o algunos de los sucedentes en ambas reglas son iguales. Entonces, se tiene:*

$$\exists R, R' \in \mathcal{R}(ant(R) \subset ant(R') \wedge suc(R) \subseteq suc(R'))$$

Algoritmo 2 Comparación de literales en las relaciones con el número de proposiciones de la BC

Entrada: La proposición que se analiza, P; el número de proposiciones de la base, $NProp$.

Salida: Valor booleano que indica si se encontraron proposiciones fuera de rango, la lista de estas proposiciones $ListaErr$.

ComparaLiterales($P, NProp, \&ListaErr$**)**
$PropAsoc \leftarrow$ Extrae_proposiciones_asociadas(P)
$Contex \leftarrow$ Extrae_contexto(P)
$EvalAlter \leftarrow$ Extrae_evaluación_alternativa(P)
for cada proposición p en la lista de proposiciones asociadas $PropAsoc$ **do**
  if $p > NProp$ **then** {Se encontró una proposición fuera de rango}
    $ListaErr \leftarrow$ Add(P)
for cada proposición p en la relación contextual $Contex$ **do**
  if $p > NProp$ **then**
    $ListaErr \leftarrow$ Add(P)
for cada proposición p en la relación de evaluación alternativa $EvalAlter$ **do**
  if $p > NProp$ **then**
    $ListaErr \leftarrow$ Add(P)
if NOT $ListaErr$ **then** {No se encontró error}
  return FALSE
else
  return TRUE

Ejemplo: En el siguiente par de reglas $R48$ está incluida en $R45$ para el sucedente 405, por tanto $R48$ puede ser omitida de la base de reglas.

$$R45: \ (218 \ \& \ 226) \ V \ (397 \mid 398) \Rightarrow 405(-0.7 \ 1);$$
$$R48: \ 226 \ \& \ 218 \Rightarrow 405(-0.7 \ 1);$$

Estas reglas pueden ser representadas por el grafo de la figura 4.2.

Definición 11 *Dos reglas R y $R' \in \mathcal{R}$ están* duplicadas *si el antecedente de R está incluido en el de R' y a su vez el antecedente de R' está contenido en R. Formalmente, se tiene:*

$$\exists R, R' \in \mathcal{R}(ant(R) \subset ant(R') \wedge ant(R') \subset ant(R))$$

Ejemplo: Las reglas $R29$ y $R30$ están duplicadas para la proposición sucedente 480. Para el resto de los sucedentes sólo el antecedente está duplicado.

$$R29: \ (473 \ \& \ -211) \ V \ (474 \ \& \ -212) \Rightarrow 480(-1 \ 0), 485(1 \ -1);$$
$$R30: \ (-212 \ \& \ 474) \ V \ (-211 \ \& \ 473) \Rightarrow 480(-1 \ 0), 481(1 \ 0.5);$$

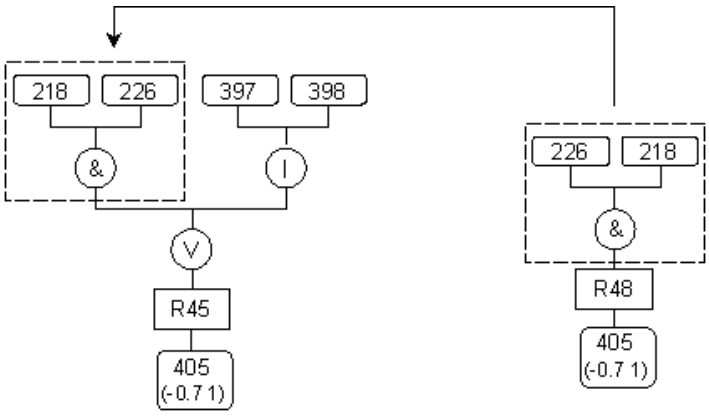

Figura 4.2: Anomalías estructurales en la base de reglas: Inclusión.

Éstas pueden rescribirse como una sola regla de la siguiente forma:

$$R29: \ (473 \ \& -211) \ V \ (474 \ \& -212) \Rightarrow 480(-1\ 0), 481(1\ 0.5), 485(1\ -1);$$

El ejemplo anterior está representado en la figura 4.3. Las proposiciones que se encuentran con doble rectángulo son proposiciones intermedios.

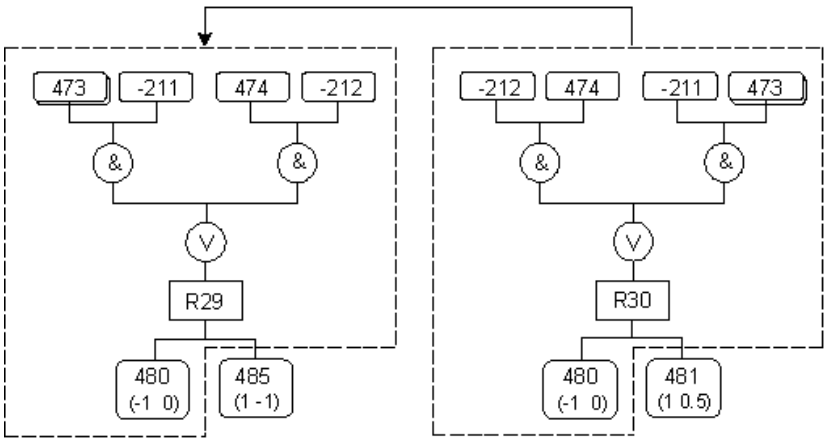

Figura 4.3: Anomalías estructurales en la base de reglas: Duplicación.

4.4.1. Métodos de detección de redundancia

Para la detección de redundancia se hace un análisis para cada par de reglas en el intervalo indicado (comenzando en $IniR$ hasta $FinR$). De esta forma, se compara la primera regla en el análisis con todas sus subsecuentes en la base (nunca con las anteriores) y así sucesivamente para el resto de las reglas. La diferencia entre inclusión y duplicación radica en la forma en que se realiza la comparación de los antecedentes de las reglas implicadas. El método general se presenta en el algoritmo 3.

Algoritmo 3 Método general de detección de redundancia

Entrada: El intervalo de verificación, $IniR, FinR$.

Salida: Valor booleano que indica si se encontró algún error; la lista de las reglas involucradas, $ListaErr$.

Redundancia($IniR, FinR, \&ListaErr$)
for cada regla $R1$ en el intervalo $IniR - FinR - 1$ **do**
  for cada regla $R2$ en el intervalo $R1 + 1 - FinR$ **do**
    if ComparaSucedentes($tipo, R1, R2, Suc$) **then** {Si hay coincidencia}
      if ComparaAntecedentes($tipo, R1, R2$) **then** {Si hay inclusión o igualdad}
        $ListaErr \leftarrow$ Add($R1$)
        $ListaErr \leftarrow$ Add($R2$)
if NOT $ListaErr$ **then** {No se encontró ningún par de reglas}
  return FALSE
else
  return TRUE

El primer paso consiste en la comparación de los sucedentes, puesto que esto requiere un análisis menos complejo y de esta forma se ejecutan menos comparaciones de antecedentes, ya que sólo se llevaran a cabo en aquellos casos donde exista coincidencia en las proposiciones sucedentes.

Para la comparación del sucedente se utiliza el algoritmo 4. Éste comienza ordenando las proposiciones sucedentes de la regla $R1$ junto con sus certidumbres correspondientes (Ordenar($SucR1$)), mismas que son almacenadas en la estructura $LSucR1$. A continuación se busca cada una de las proposiciones de $R2$ en la lista ordenada (BuscaSucedente). A esta función de búsqueda se la transfieren además los grados de certidumbre (WS_{i1} WS_{i2}) de la proposición correspondiente para la coincidencia (en caso de inclusión, duplicación y condiciones innecesarias) o desigualdad (en caso de contradicción), lo cual se indica con el parámetro *tipo* ($'I'$ para igualdad en las certidumbres y $'C'$ para contradicción).

El segundo paso del algoritmo 3 consiste en el estudio del antecedente para aquellas reglas que presenten coincidencias en los sucedentes. En este punto se realiza la descomposición del mismo para realizar la comparación. Para esto es necesario identificar algunos elementos. Por ejemplo, en la siguiente regla:

$$R1: \ (4\ V\ 5)\ \&\ (1\ V\ 2)\ \&\ (7\ V\ 8\ V\ 9)\ \&\ (3 \mid 5 \mid 6 \mid 10);$$

El conectivo principal es & y la regla está formada por cuatro partes: $(4\ V\ 5), (1\ V\ 2)$, $(7\ V\ 8\ V\ 9)$ y $(3 \mid 5 \mid 6 \mid 10)$. Los conectivos secundarios que vinculan cada parte son: $V, V, V, \mid$ respectivamente.

Algoritmo 4 Comparación de los sucedentes de dos reglas

Entrada: El tipo de comparación, *tipo*; las dos reglas a comparar, $R1, R2$.

Salida: Valor booleano que indica si hay coincidencias de sucedentes; las proposiciones sucedentes que coinciden, *Suc*

ComparaSucedentes(*tipo*, $*R1, *R2, \&Suc$**)**
$SucR1 \leftarrow$ Extrae_sucedente($R1$)
$SucR2 \leftarrow$ Extrae_sucedente($R2$)
$LSucR1 \leftarrow$ Ordenar($SucR1$)
for cada proposición p sucedente en $SucR2$ **do**
  $ind \leftarrow$BuscaSucedente($tipo, LSucR1, p, Cert1(p), Cert2(p)$)
  if $ind >= 0$ **then** {Se encontró}
    $Suc \leftarrow$ Add(p)
if NOT Suc **then** {No se encontraron coincidencias en los sucedentes}
  return FALSE
else
  return TRUE

El algoritmo de comparación de antecedentes se presenta en el algoritmo 5. El *tipo* en el algoritmo indica la forma en que se realiza la comparación. Cuando *tipo* es igual a $'I'$ indica que se buscan antecedentes incluidos (inclusión) y cuando es igual a $'D'$ se buscan antecedentes iguales (duplicación).

Algoritmo 5 Comparación de los antecedentes de dos reglas

Entrada: El tipo de comparación, *tipo*; las dos reglas a comparar, $R1, R2$.

Salida: Valor booleano que indica si las reglas son iguales o una está incluida en la otra (según *tipo*)

ComparaAntecedentes(*tipo*, $*R1, *R2$**)**
if $tipo = 'D'$ **then** {Se comprueba que los antecedentes sean iguales}
  if numPartes($R1$) <> numPartes($R2$) OR conectPpal($R1$) <> conectPpal($R2$) **then**
    return FALSE
else {Se comprueba si uno está incluido en el otro ($tipo = 'I'$)}
  if numPartes($R1$) = numPartes($R2$) **then**
    return FALSE
if ComparaPartes($R1, R2$) **then** {Se no se encuentra correspondecia entre partes}
  return FALSE
return TRUE

En el caso de antecedentes incluidos se analiza primeramente que el número de partes de las reglas (numPartes(R)) no sean iguales (para que una pueda estar incluida en la otra), por el contrario, en el caso de duplicación se comprueba que el número de

partes y el conectivo principal de ambas reglas sean los mismos.

El estudio de la correspondencia entre las partes se realiza en dos los casos en ComparaPartes. En esta función se examina que las proposiciones y el conectivo secundario que las relacionan sean los mismos entre pares de partes de ambas reglas.

4.5. Circularidad

Definición 12 *Una BC contiene* dependencia circular local *en su base de reglas si existe una hipótesis $I \in \mathcal{I}$ que forma parte del sucedente de una regla $R \in \mathcal{R}$ y ésta sólo puede ser evaluada cuando es dada como entrada a $\mathcal{R}$ o inferida a partir de otras reglas. Formalmente, una BC contiene circularidad si:*

$$\exists I \in \mathcal{I} \subset \mathcal{P},\ R \in \mathcal{R}, E \in \mathcal{E}$$
$$(I \subset suc(R) \wedge \neg infiere(I, \mathcal{R}, E) \wedge infiere(I, \mathcal{R}, E \cup \{I\}))$$

Esto es, I depende de él mismo. Si el valor de I no fuera conocido, entonces no habría forma de inferirlo.

Ejemplo: La próxima regla contiene un ciclo local formado por la proposición 473, dado que ésta se encuentra tanto en el antecedente como en el sucedente de $R60$. Su representación como grafo se muestra en la figura 4.4

$$R60: \ (\mathbf{473}\ \&\ 211)\ V\ (474\ \&\ 212)\ V\ (475\ \&\ 240) \Rightarrow \mathbf{473}(0.5\ \ -1), 480(1\ 0)$$

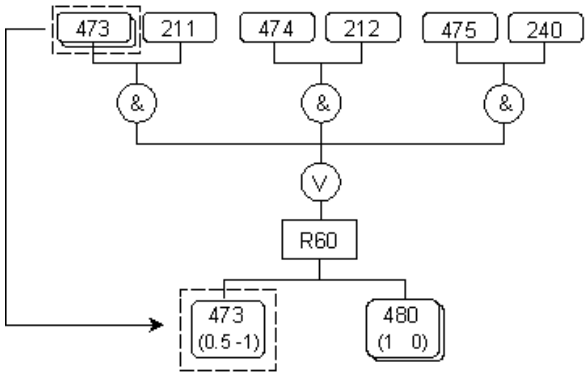

Figura 4.4: Anomalías estructurales en la base de reglas: Ciclo local.

En la parte condicional de las restricciones también se puede encontrar circularidad.

Definición 13 *Una BC contiene* dependencia circular local *en su base de proposiciones si existe una proposición $P \in \mathcal{P}$ que posee una restricción semántica $C \in \mathcal{C}$,*

donde la condición de C puede ser evaluada sólo cuando P es dada como entrada a $\mathcal{C}$ o inferida a partir de las reglas. Formalmente, una BC contiene circularidad local si:

$$\exists P \in \mathcal{P},\, C \in \mathcal{C}, E \in \mathcal{E}$$
$$(P \subset cond(C) \wedge \neg infiere(P, \mathcal{C}, E) \wedge infiere(P, \mathcal{C}, E \cup \{P\}))$$

Ejemplo: La relación contextual contiene un ciclo local formado por la proposición 154, ya que ésta forma parte de la condición. La figura 4.5 representa la relación.

$$RC154: \ \mathbf{154}(0.3) \ @ \ 153 \ \& \ \mathbf{154}$$

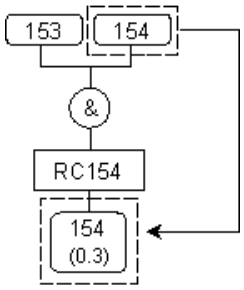

Figura 4.5: Anomalías estructurales en la base de proposiciones: Ciclo local.

Definición 14 *Una BC contiene* dependencia circular particular *en su base de reglas si existen dos hipótesis $I, I' \in \mathcal{I}$ que son sucedentes de las reglas $R, R' \in \mathcal{R}$ respectivamente, donde el sucedente de R (I) se puede inferir sólo cuando el valor de I' es dado como entrada a $\mathcal{R}$ o evaluado a partir de otras reglas y el sucedente de R' (I') se puede inferir sólo cuando se conoce I . Formalmente, una BC contiene circularidad particular si:*

$$\exists I, I' \in \mathcal{I} \subset \mathcal{P}, R, R' \in \mathcal{R}, E \in \mathcal{E}$$
$$(I \subset suc(R) \wedge \neg infiere(I, \mathcal{R}, E) \wedge infiere(I, \mathcal{R}, E \cup \{I'\}) \wedge$$
$$(I' \subset suc(R') \wedge \neg infiere(I', \mathcal{R}, E) \wedge infiere(I', \mathcal{R}, E \cup \{I\}))$$

Este es un caso específico de ciclo particular en el que dos reglas están involucradas $R : I' \Rightarrow I$ y $R' : I \Rightarrow I'$, aunque también se pueden formar lazos que impliquen más de dos reglas, por ejemplo con tres tenemos:

$$R: \ I' \Rightarrow I$$
$$R': \ I'' \Rightarrow I'$$
$$R'': \ I \Rightarrow I''$$

Y así sucesivamente para n reglas involucradas.

Ejemplo: En el siguiente conjunto de reglas se forma un ciclo particular. La figura 4.6 representa dicho conjunto.

$$R75: \ (479 \ \& \ 189) \ V \ (157 \ \& \ 190) \Rightarrow \mathbf{477}(1 \ 0), 478(-1 \ 0);$$
$$R76: \ (191 \mid 185) \ \& \ (192 \ V \ -188) \Rightarrow 479(-1 \ 0);$$
$$R77: \ 395 \ \& \ (\mathbf{477} \ V \ 186) \ \& \ (478 \ V \ 187) \Rightarrow 185(1 \ 0);$$

Como se mencionó HAries implementa encadenamiento hacia atrás, enfocando la atención en un objetivo (sucedente) y pidiendo información para confirmarlo o negarlo (en el antecedente). Haciendo este análisis para $R75$, cuando se trata de evaluar la proposición 477, se desencadena la evaluación de 479 en $R76$, y así sucesivamente hasta volver a encontrar a 477 en $R77$.

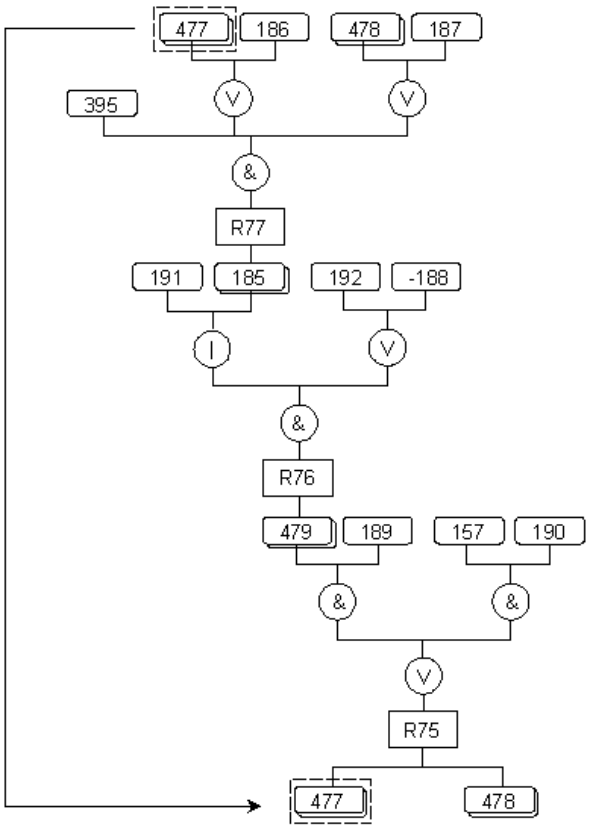

Figura 4.6: Anomalías estructurales en la base de reglas: Ciclo particular.

Definición 15 *Una BC contiene* dependencia circular particular *en su base de proposiciones si existen dos proposiciones $P, P' \in \mathcal{P}$ que forman parte de las restricciones semánticas $C, C' \in \mathcal{C}$ respectivamente, donde la condición de C (P) se puede evaluar sólo cuando P' es dada como entrada a $\mathcal{C}$ o inferida a partir de las reglas y la condición de C' (P') se puede evaluar sólo cuando se conoce P. Formalmente, una BC contiene*

circularidad particular si:

$$\exists, P, P' \in \mathcal{P}, C, C' \in \mathcal{C}, E \in \mathcal{E}$$
$$(P \subset cond(C) \wedge \neg infiere(P, \mathcal{C}, E) \wedge infiere(C, \mathcal{C}, E \cup \{P'\})$$
$$(P' \subset cond(C') \wedge \neg infiere(P', \mathcal{C}, E) \wedge infiere(P', \mathcal{C}, E \cup \{P\}))$$

Este es un caso específico de ciclo particular en el que dos restricciones están involucradas $C : P(x) \ @ \ P'$ y $C' : P'(y) \ @ \ P$, aunque también se pueden formar lazos que impliquen más de dos restricciones, por ejemplo con tres tenemos:

$$C : \ P(x) \ @ \ P'$$
$$C' : \ P'(y) \ @ \ P''$$
$$C'' : \ P(z)'' \ @ \ P$$

Donde x, y, z son los pesos alternos de las relaciones contextuales.

Ejemplo: En el siguiente conjunto de restricciones se forma un ciclo particular. La representación de este conjunto se muestra en 4.7.

$$RC33 : \ \mathbf{33}(0.5) \ @ \ 34 \ \& \ 36$$
$$RC34 : \ 34(-0.3) \ @ \ 35 \ V \ 37$$
$$RC35 : \ 35(1.0) \ @ \ \mathbf{33} \ \& \ 38$$

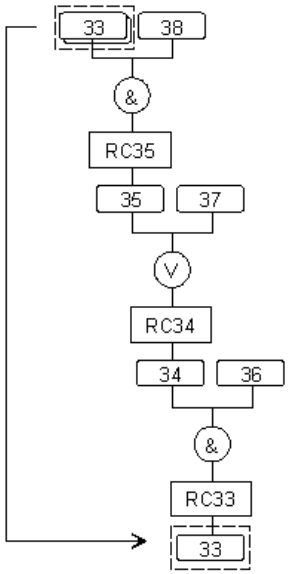

Figura 4.7: Anomalías estructurales en la base de proposiciones: Ciclo particular.

Cuando se va a inferir el valor de certidumbre de la proposición 33, ésta tiene asociada una relación contextual ($RC33$), por tanto el sistema procede con la evaluación

de la condición, en la cual se encuentra la proposición 34. Esto desencadena la evaluación de la proposición 35 en la condición de $RC34$ y así sucesivamente hasta encontrar a 33 en $RC35$, formándose entonces un ciclo particular.

Definición 16 *Una BC contiene* dependencia circular global *si existen dos proposiciones $P, P' \in \mathcal{P}$ donde P es sucedente de la regla $R \in \mathcal{R}$ y forma parte de la condición en una restricción $C \in \mathcal{C}$ asociada a la hipótesis P'; entonces el sucedente de R (P) se puede inferir sólo cuando se conoce el valor de P' y a su vez P' se puede inferir sólo cuando ha sido evaluada la condición de C a la cual pertenece P. Formalmente, una BC contiene circularidad global si:*

$$\exists R \in \mathcal{R}, C \in \mathcal{C}, E \in \mathcal{E}, P, P' \in \mathcal{P}$$
$$(P \subset suc(R) \wedge \neg infiere(P, \mathcal{R}, E) \wedge infiere(P, \mathcal{R}, E \cup \{P'\})$$
$$(P \subset cond(C) \wedge \neg infiere(P', \mathcal{C}, E) \wedge infiere(P', \mathcal{C}, E \cup \{P\}))$$

La anterior es una instancia particular de un ciclo global en el que hay una regla y una restricción involucradas $R : P' \Rightarrow P$ y $C : P'(x)$ @ P, donde x el peso alterno de la relación contextual. De la misma forma que en los ciclos particulares, aquí pueden estar involucrados más elementos, por ejemplo:

$$R: \ P' \Rightarrow P$$
$$C: \ P'(x) \ @ \ P''$$
$$R': \ P \Rightarrow P''$$

Y así sucesivamente para n elementos.

Ejemplo: En el siguiente conjunto se establece un ciclo global que involucra reglas y proposiciones (mediante sus relaciones). La figura 4.8 muestra la representación de dicho conjunto.

$$R92: \ (464 \ \& - 181 \ \& - 182) \ V \ (465 \ \& - 183 \ \& \ 252) \Rightarrow \mathbf{477}(-1 \ 0);$$
$$RC464: \ 464(0.7) \ @ \ 476$$
$$EA476: \ 476 \ \xi \ 478 \ \& \ 479 \ \& \ 480$$
$$R93: \ \mathbf{477} \mid 126 \Rightarrow 478(-3 \ 5);$$

Cuando se necesita inferir el valor de la proposición 477 se desencadena la evaluación de la regla $R92$, en cuyo antecedente se encuentra la proposición 464, pero ésta tiene una relación contextual que establece como condición el cumplimiento de 476; por tanto se necesita conocer la certidumbre de la proposición 476. La evaluación de 476 provoca que se analice la condición de la relación de evaluación alternativa que incluye las proposiciones 478 y 479. Finalmente, cuando se trata de evaluar a 478 en $R93$ encontramos nuevamente a 477, proposición de la cual partimos, en el antecedente de la regla. De esta forma nunca puede encontrarse el valor de 477 por el ciclo generado durante su estimación.

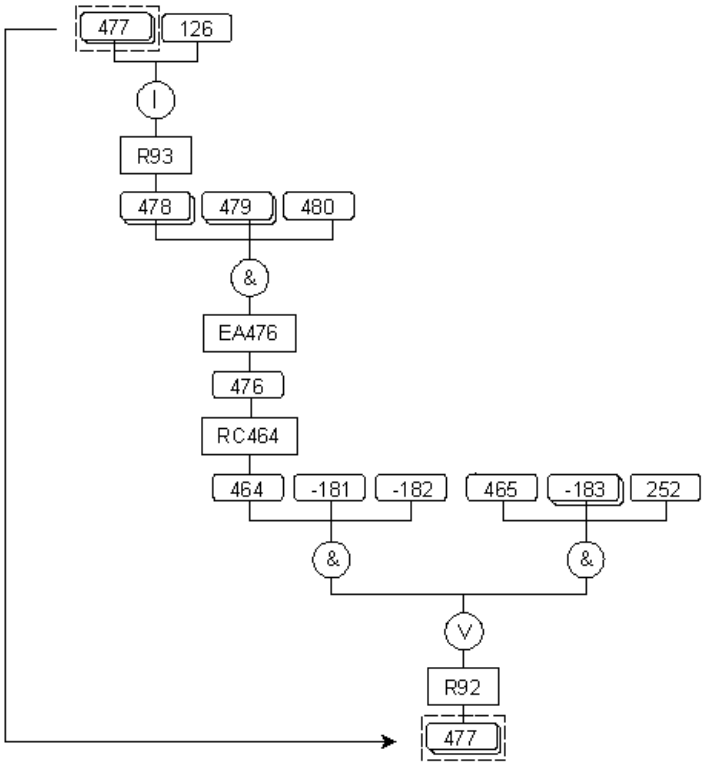

Figura 4.8: Anomalías estructurales en la base de reglas y de proposiciones: Ciclo global.

4.5.1. Métodos de detección de circularidad

Para la detección de ciclos locales, tanto en la base de reglas como en la de proposiciones, se hace un recorrido de los literales que forman parte del antecedente de la regla o de las relaciones en las proposiciones. Los procedimientos que se siguen son muy similares a los empleados para la localización de valores ilegales, con la única diferencia que las comparaciones se realizan con las proposiciones de la acción (en vez del total de proposiciones de la BC) buscando encontrar coincidencias para reportar entonces la formación de un lazo.

Para la detección de ciclos particulares en la base de reglas se emplea el método del algoritmo 6. De manera general el algoritmo parte de la primera proposición en la base y localiza las reglas en las cuales dicha proposición forma parte del sucedente, y comienza hacer el recorrido por éstas reglas como se realizara en el proceso de inferencia.

Aquí es necesario aclarar que el algoritmo de detección no hace este análisis para todas las proposiciones que forman parte del sucedente, como se especificó en las definiciones sólo se toman en cuenta aquellas proposiciones que son intermedios, puesto que son las únicas que aparecen en ambas partes de la regla.

Algoritmo 6 Método de detección de ciclos particulares en la base de reglas

Entrada: La regla que se analiza, R; la pila donde se almacenan las proposiciones analizadas, $Pila$.

Salida: Valor booleano que indica si se encontró o no un ciclo, $Ciclo$.

```
RPGCiclos(R, &Pila)
Ciclo ← FALSE
*Ant ←Extrae_antecedente(R)
for cada proposición p en el antecedente Ant do
  if Busca(p, Pila) then {Se encontró un ciclo}
    return TRUE
for cada proposición p en el antecedente Ant do
  TipoPro ← Determina_tipo_proposición(p)
  if TipoPro = 'Intermedio' then
    Pila ← Push(p)
    Rpg ← Saca_1raRegla_esSucedente(p)
    while Rpg > 0 AND NOT Ciclo do
      Pila ← Push(Rpg)
      Ciclo ← RPGCiclos(Rpg, Pila)
      if NOT Ciclo then {Saca la regla de la pila}
        Pila ← Pop( )
        Rpg ← Proxima_regla_esSucedente(p, Rpg)
      else
        return TRUE
    if NOT Ciclo then {Saca la proposición de la pila}
      Pila ← Pop( )
return FALSE
```

En el algoritmo de detección, una vez que se ha verificado que la proposición de la cual parte el análisis es intermedio, se examina cada regla en la cual está dicha proposición como parte del sucedente. En el estudio de las reglas primero se obtienen las proposiciones que forman parte del antecedente y se verifica que no hayan sido introducidas a una pila que almacena las proposiciones ya analizadas, si se encuentra alguna de las proposiciones antecedentes, esto indica que se ha formado un ciclo. Si no se encuentra ninguna se inicia el análisis con la primera proposición antecedente (identificada por p en el código) en caso de que ésta sea intermedio ($TipoPro = 'Intermedio'$), introduciéndola en la pila ($Pila \leftarrow \text{Push}(p)$) y obteniendo la 1ra regla donde se encuentre como parte del sucedente (1raRegla_conSucedente(p)). Si se encuentra ($Rpg > 0$, Rpg indica el número o identificador de la regla en la base), se añade la regla a la pila ($Pila \leftarrow \text{Push}(Rpg)$) y se llama al algoritmo de manera recursiva con la regla extraída. Si no se encuentra ningún ciclo (NOT $Ciclo$) con la regla analizada, se saca ésta de la pila ($Pila \leftarrow \text{Pop}()$) y se extrae la próxima regla de la base con la proposición p como sucedente (Próxima_regla_esSucedente(p)). Se realiza el mismo análisis mientras

se continúe encontrando a p en el sucedente de alguna regla. Si no se encuentra, se saca a la proposición p de la pila y se continúa con la próxima proposición del antecedente.

La detección de ciclos en la base de proposiciones se presenta en el algoritmo 7 y se realiza de forma muy parecida a las reglas. La diferencia aquí radica en que el recorrido se realiza por las relaciones definidas para cada proposición.

Algoritmo 7 Método de detección de ciclos particulares en la base de proposiciones

Entrada: La proposición que se analiza, P; la pila donde se almacenan las proposiciones analizadas, $Pila$.

Salida: Valor booleano que indica si se encontró o no un ciclo, $Ciclo$.

```
PropCiclos(P, &Pila)
*Literales ←Extrae_relaciones(P)
for cada proposición p en la lista Literales do
  if Busca(p, Pila) then {Se encontró un ciclo}
    return TRUE
for cada proposición p en la lista Literales AND NOT Ciclo do
  Pila ← Push(p)
  Ciclo ← PropCiclos(p, Pila)
  if NOT Ciclo then {Saca la proposiciónde la pila}
    Pila ← Pop( )
  else
    return TRUE
return FALSE
```

El primer paso consiste en la obtención de los literales que forman parte de todas las posibles relaciones de la proposición en cuestión ($Literales$) y la verificación de que no hayan sido introducidas a la pila. Si no se encuentra ningún ciclo se inicia el análisis con la 1ra proposición del conjunto de literales (identificada por p en el código), introduciéndola en la pila ($Pila \leftarrow \text{Push}(p)$) y se llama al algoritmo de manera recursiva con la proposición actual. Si no se encuentra ningún ciclo (NOT $Ciclo$) con la proposición analizada, se saca ésta de la pila ($Pila \leftarrow \text{Pop}()$) y se continúa con la próxima proposición del conjunto.

4.6. Inconsistencia

Definición 17 *Un par de reglas $R, R' \in \mathcal{R}$ son* contradictorias *si el antecedente de R' incluye o duplica al antecedente de R y la evaluación de alguno de sus sucedentes provocan una inconsistencia en cuanto a la contribución en los pesos del mismo. Entonces formalmente R y R' derivan en una contradicción si:*

$$\exists R, R' \in \mathcal{R}(ant(R) \subseteq ant(R') \wedge suc(R)_i = suc(R')_j \wedge cert(suc(R)_i) \neq cert(suc(R')_j))$$

Ejemplo: El siguiente par de reglas $R25$ y $R26$ presentan una contradicción para la proposición sucedente 478, puesto que para una misma condición las contribuciones en los pesos son diferentes. La representación de estas reglas se presenta en la figura 4.9

$$R25: \ 471 \ V \ (209 \mid 258) \ V \ (325 \ \& \ 384) \Rightarrow \mathbf{478}(0.5\ 0), 479(1\ 0), 480(-1\ 0);$$
$$R26: \ (209 \mid 258) \Rightarrow 477(0.8\ 1), \mathbf{478}(-1\ 1);$$

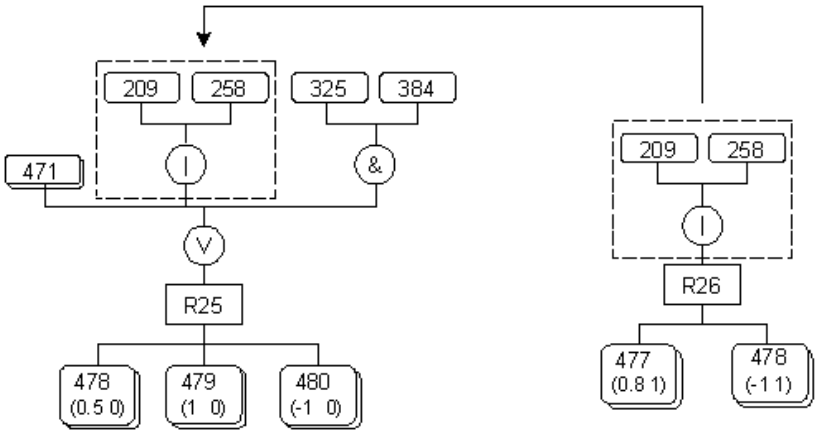

Figura 4.9: Anomalías lógicas en la base de reglas: Contradicción.

Definición 18 *Un par de reglas $R, R' \in \mathcal{R}$ contienen una* condición innecesaria *si los sucedentes de las reglas son iguales y los antecedentes sólo se diferencian en que alguno de los literales que forman parte de R está negado en R'. Formalmente, R y R' tienen una condición innecesaria si:*

$$\exists R,R' \in \mathcal{R}$$
$$(suc(R) = suc(R') \wedge ant(R) - \{ant(R)_i\} = ant(R') - \{ant(R')_j\}) \wedge$$
$$(ant(R)_i = \neg ant(R')_j)$$

Ejemplo: Las reglas $R12$ y $R37$, representadas en la figura 4.10, contienen una condición innecesaria para la proposición 212, puesto que el cumplimiento (o incumplimiento) de la misma no afecta a los sucedentes ni a la contribución de los pesos.

$$R12: \ \mathbf{212} \ \& \ 135 \ \& \ 234 \Rightarrow 179(-1\ 1), 183(-0.3\ 0.5);$$
$$R37: \ 135 \ \& \ \mathbf{-212} \ \& \ 234 \Rightarrow 179(-1\ 1), 183(-0.3\ 0.5);$$

Las reglas anteriores pudieran ser resumidas en una sola:

$$R12: \ 135 \ \& \ 234 \Rightarrow 179(-1\ 1), 183(-0.3\ 0.5);$$

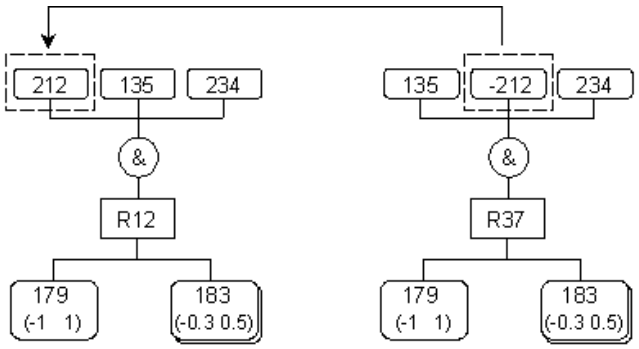

Figura 4.10: Anomalías lógicas en la base de reglas: Condiciones innecesarias.

4.6.1. Métodos de detección de inconsistencia

El método general que se utiliza para la detección de inconsistencias es básicamente el mismo del algoritmo 3 para la redundancia. En el caso de contradicción para el análisis de los sucedentes se buscan que los pesos sean diferentes (se coloca $tipo = 'C'$) y en la comparación del antecedente se buscan reglas incluidas ($tipo = 'I'$). Para las condiciones innecesarias se buscan sucedentes con los mismos pesos ($tipo = 'I'$) y reglas duplicadas ($tipo = 'D'$).

4.7. Estudio empírico

En esta sección se muestran los resultados de aplicar los algoritmos descritos en los epígrafes anteriores a un conjunto de SBC empleados en casos prácticos para diversos dominios del conocimiento. Con este fin se empleó VBCH para verificar las BC de los sistemas especificados en el cuadro 4.1 y de los cuales se presenta una breve descripción a continuación.

SBC para el despacho de carga eléctrica

El despacho de carga eléctrica, representa un problema donde se mezclan aspectos establecidos sobre como operar por una parte y la experiencia para la toma de decisiones en un momento dado por otra. Son típicas las situaciones en las que un operador debe ejecutar un conjunto de operaciones bastante grande de forma tal que garanticen un mínimo de interrupciones. Así por ejemplo, están los casos donde se necesite brindar un mantenimiento, distribuir de forma conveniente la carga, efectuar cambios en el suministro de la energía de una fuente a otra, etc.

Tomando en cuenta estos hechos se construyó un sistema que fuera capaz de asesorar el trabajo de estos operadores, buscando así la disminución de los posibles errores [31]. La BC de este sistema recibe el nombre de TRANSFOR y será la primera en ser considerada dentro del análisis comparativo que se realiza más adelante.

SBC para la enseñanza de los métodos de análisis exploratorio de datos

El análisis exploratorio de datos constituye un área de investigación, cuyos métodos permiten resolver problemas de carácter observacional de muy variada naturaleza y complejidad. Se incluyen como parte de éstos, técnicas para reconocimiento de patrones, generación automatizada de hipótesis, selección de variable, reducción de la dimensionalidad, etc.

Ocurre sin embargo, que las personas con problemas en sus áreas de trabajo que pudieran ser resueltos satisfactoriamente utilizando estas aproximaciones, por lo general son desconocedoras de ellas o presentan un nivel de información que no les permite usarlas.

Por ello se procedió al diseño y construcción de un sistema especializado en ayudar a los usuarios, no sólo para decidir que hacer ante una situación dada, sino también para enseñar la teoría y práctica necesarias para poder hacer un uso efectivo de ellos [26]. El estudio comenzó con la elaboración de la BC ESFEDA.

SBC para el pronóstico en pozos de petróleo

Este sistema se desarrolla en apoyo a un experimento consistente en la inyección de tenso-activos a un conjunto de pozos, con el objetivo de aumentar sus producciones y es el resultado de un estudio de las condiciones bajo las cuáles es posible pronosticar un efecto positivo o negativo antes de proceder a la activación del pozo en análisis.

Como información de partida para buscar respuesta a esta interrogante, se utilizó la caracterización del grupo de pozos estudiados a través de un conjunto de variables, que se relacionan con las características del crudo, la geología, la tecnología empleada, las características del caldo (tenso-activo) y el efecto obtenido [32]. La BC de este sistema recibió el nombre de POZOS.

Sistema para el diagnóstico y tratamiento de la epilepsia

EPILEP es una base de conocimientos, que junto con la máquina de inferencias de HAries constituyen un SBC cuyo objetivo central es servir como consultante a médicos que se enfrenten al diagnóstico y tratamiento de la epilepsia, el cual constituye uno de los problemas más complejos en neurología.

El análisis incluye: el diagnóstico diferencial, donde se debe establecer si las crisis son de naturaleza epiléptica o no, la clasificación clínica de las crisis, la clasificación de las epilepsias, el estudio etiológico, el tratamiento y el pronóstico [33].

BC	# Reglas	# Proposiciones	Propósito
TRANSFOR	69	93	Despacho de carga eléctrica
ESFEDA	110	325	Enseñanza de los métodos de análisis exploratorio de datos
POZOS	116	85	Pronóstico de la inyección de tenso - activos en pozos de petróleo
EPILEP	943	862	Diagnóstico y tratamiento de epilepsia

Cuadro 4.1: Características de las BC

Las cantidades de ciclos de reloj obtenidos tras ejecutar VBCH para las BC de los cuatro sistemas descritos aparecen en el cuadro 4.2. Estos datos fueron obtenidos corriendo la herramienta en una Pentium 4 a 2.66 GHz.

Anomalía	TRANSFOR	ESFEDA	POZOS	EPILEP
Valores ilegales	0	0	0	0
Redundancia				
Inclusión	0	0	10	311
Duplicación	10	10	20	841
Circularidad				
Local	0	0	0	0
Particular	0	0	0	611
Global	0	0	0	648
Inconsistencia				
Contradicción	0	0	10	341
Condiciones innecesarias	0	0	10	384

Cuadro 4.2: Ciclos de reloj en la ejecución de los métodos de detección de anomalías

En los valores ilegales y ciclos locales, como se hace un análisis lineal de cada regla, no se alcanzan a apreciar aumentos en las cantidades, independientemente del tamaño de la base.

El costo principal en los casos de redundancia e inconsistencia se debe a la comparación de los antecedentes entre pares de reglas. La búsqueda de sucedentes iguales o contradictorios es una operación menor, y es por ello que el costo del chequeo depende del número de reglas que posean sucedentes con las característica mencionadas. Esta es la razón por la cual el tiempo que se requiere para la detección de éstas anomalías en la base POZOS es mayor que el de ESFEDA, aún cuando éstos poseen casi la misma cantidad de reglas.

En los casos de circularidad la experiencia y las evidencias actuales sugieren que las cadenas de inferencia cortas son mucho más comunes en la práctica que las extensas. Siendo este el caso de las bases analizadas.

El cuadro 4.3 muestra el número de anomalías de cada tipo detectadas por VBCH. En todos los casos se tomó como intervalo de verificación la BC completa.

Anomalía	TRANSFOR	ESFEDA	POZOS	EPILEP
Valores ilegales	1	3	0	0
Redundancia				
Inclusión	0	0	1	3
Duplicación	1	2	2	8
Circularidad				
Local	0	0	3	0
Particular	0	0	4	0
Global	0	0	7	0
Inconsistencia				
Contradicción	0	0	0	1
Condiciones innecesarias	0	0	0	2

Cuadro 4.3: Número de anomalías detectadas por caso de estudio

El resultado más llamativo aquí es que a pesar de los esfuerzos de los diseñadores, las BC de sistemas reales casi siempre contienen alguna anomalía, aunque en la mayoría de los casos, estas no conlleven a errores en la ejecución.

Los problemas más comunes para los sistemas son de redundancia y específicamente de duplicación de los antecedentes entre pares de reglas.

Los ciclos en la base POZOS fueron intencionalmente introducidos para probar de manera explícita el razonamiento con lazos, al igual que los valores ilegales en ESFEDA y TRANSFOR.

4.8. Conclusiones

En este capítulo se ha presentado la clasificación y definición de las anomalías que pueden ser detectadas por el sistema VBCH para las dos estructuras principales de representación del conocimiento del lenguaje HAries: proposiciones y reglas de producción.

Por otra parte, para cada una de las anomalías identificadas se ha presentado un ejemplo que describe la forma en que se presentan las mismas en la BC y los algoritmos que se implementaron para detectarlas.

Finalmente se presentó una discusión sobre las consideraciones de desempeño y utilidad de los algoritmos, basada un estudio empírico en el que se analizaron cuatro BC construidas para sistemas de aplicación real en varios dominios del conocimiento.

Capítulo 5

Análisis de Coeficientes

Además de la detección de anomalías en el proceso de verificación, como complemento de la estructura de las BC en HAries, resulta muy conveniente contar con ciertas medidas para evaluar la configuración del conocimiento representado.

La idea radica en definir algunos coeficientes, que además de brindar datos sobre la organización, también se utilicen para advertir sobre las posibles deficiencias que pudieran afectar la ejecución exitosa de un sistema. Estas medidas están orientadas al análisis de las bases de proposiciones y reglas, por tanto, todos los conceptos que se emplean en la definición formal de los coeficientes están dentro de este contexto.

5.1. Conceptos previos

Definición 19 *Sea $R \in \mathcal{R}$ un conjunto no vacío de reglas cualesquiera, se dice que una secuencia $R_1, ..., R_k$ de elementos de R es una* Cadena o Línea de Razonamiento *si se cumple que para cada $R_i (i = 1, ..., k-1)$ alguna de sus proposiciones sucedentes aparece también como parte del antecedente de $R_i + 1$. Formalmente, R es una línea de razonamiento si:*

$$\exists R_i, R_{i+1} \in \mathcal{R}, P \in \mathcal{P}(P \subset suc(R_i) \wedge P \subset ant(R_{i+1}))$$

Definición 20 *Sea L_i una línea de razonamiento cualquiera formada por $R_1, ..., R_k$ reglas. Se dice que L_i está* asociada *a la proposición P_j si se cumple que P_j aparece como sucedente de la regla R_k y se denota $L_i P_j$.*

$$\exists R_1, ..., R_k \in \mathcal{R}, P_j \in \mathcal{P}(R_1, ..., R_k \subseteq L_i \wedge P_j \subset suc(R_k))$$

Definición 21 *Se define como* profundidad *de una línea de razonamiento L_i formada por $R_1, ..., R_k$ reglas (se denota PL_i) a la cantidad de niveles intermedios que existen entre los sucedentes de R_k y el antecedente de R_1, es decir: $PL_i = k - 1$.*

Definición 22 *Sea P_i una proposición pregunta o intermedio de una base de proposiciones $\mathcal{P}$, se dice que ésta es* accesible *desde otra proposición intermedio u objetivo P_j si existe una línea de razonamiento $R_1, ..., R_k$ de reglas tal que P_i ocurre en el antecedente de R_1 y P_j es sucedente de R_k. Formalmente,*

$$\exists P_i \in \mathcal{Q} \cup \mathcal{I}, P_j \in \mathcal{I} \cup \mathcal{O} \subset \mathcal{P}, R_1, ..., R_k \in \mathcal{R}(P_i \subset ant(R_1) \wedge P_j \subset suc(R_k))$$

Definición 23 *Se llama* Base Informativa *asociada a la proposición intermedio u objetivo P_j, al conjunto QP_j de proposiciones preguntas ($QP_j \subseteq \mathcal{Q}$) accesibles desde P_j.*

Definición 24 *Sea $\mathcal{O}$ el conjunto de proposiciones objetivos una base de proposiciones $\mathcal{P}$, se dice que una proposición P_i cualquiera es* evidencia directa *de un objetivo de P_j, que aparece en el sucedente de un conjunto de reglas $R_1, ..., R_k$, si se encuentra como parte de los antecedentes de dichas reglas.*

$$\exists P_i \in \mathcal{P}, R_1, ..., R_k \in \mathcal{R}(P_i \subset ant(R_1) \cup ... \cup ant(R_k) \wedge P_j \subset suc(R_i))$$

5.2. Clasificación y definición de coeficientes

Los coeficientes para el estudio de las BC descritas con HAries han sido clasificados en dos categorías, como se muestra en la figura 5.1.

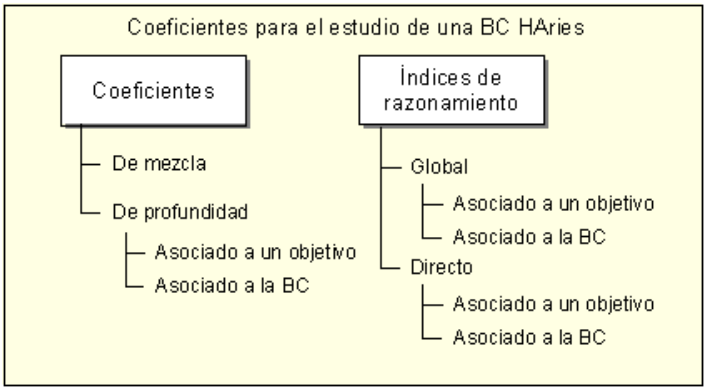

Figura 5.1: Clasificación de los coeficientes en HAries.

- El *coeficiente de mezcla* mide la coincidencia entre las bases informativas de todos los objetivos de la BC.
- El *coeficiente de profundidad* calcula los niveles en las líneas de razonamiento asociadas a una proposición objetivo en particular o todos los objetivos de la BC.
- El *Índice de razonamiento global* determina la proporción entre las proposiciones preguntas e intermedios accesibles de uno o todos los objetivos de la base de reglas.

- El *Índice de razonamiento directo* obtiene la proporción entre las proposiciones preguntas e intermedios que son evidencias directas de los objetivos.

5.3. Coeficiente de mezcla

El concepto de independencia entre objetivos está relacionado, en cierto sentido, con el grado de complejidad existente en la BC, puesto que mientras mayor sea el grado de solapamiento de las bases informativas asociadas a los objetivos, mayor dificultad presenta el problema para su representación, ya que se habla de partir de las mismas evidencias, para llegar a resultados diferentes.

Para medir este tipo de complejidad cuantitativamente se puede definir un coeficiente asociado a una BC como sigue:

Definición 25 *Se llama* Coeficiente de Mezcla *(Cz) de una BC en relación con las proposiciones objetivos y las reglas a la expresión:*

$$Cz = \frac{2 \times \sum_{i=1}^{No-1} \sum_{j=i+1}^{No} card(QP_i \cap QP_j)}{\sum_{i=1}^{No-1} \sum_{j=i+1}^{No} Min\{card(QP_i), card(QP_j)\}} - 1$$

donde No es el número de objetivos,
QP_i y QP_j son las bases informativas de los objetivos i y j respectivamente y
card representa el cardinal del conjunto indicado.

La idea de este coeficiente radica en comparar todos los objetivos dos a dos para contar las coincidencias de sus bases informativas. Esto brinda un resultado entre -1 y 1 donde -1 indica la no existencia de mezclas entre los objetivos, es decir, que son independientes dos a dos y 1 que todas las bases informativas coinciden y son igual al cuestionario.

Ejemplo: Tomemos las siguientes cadenas de razonamiento para las proposiciones objetivos: 36 y 38, cuya representación se muestra en la figura 5.2. Las proposiciones intermedios están representadas con doble rectángulo.

$R11:\ 12\ \&\ 13\ \&14 \Rightarrow 27(-1\ 1);$ $R12:\ 14\ \&\ 15 \Rightarrow 29(0\ 1)$
$R18:\ 16\ V\ 27 \Rightarrow 32(-0.5\ 1), 35(0\ 0.3);$ $R19:\ 17\ \&\ 29 \Rightarrow \mathbf{38}(-1\ 0.5)$
$R23:\ 17\ \&\ (32 \mid 35) \Rightarrow \mathbf{36}(-0.2\ 0.4)$

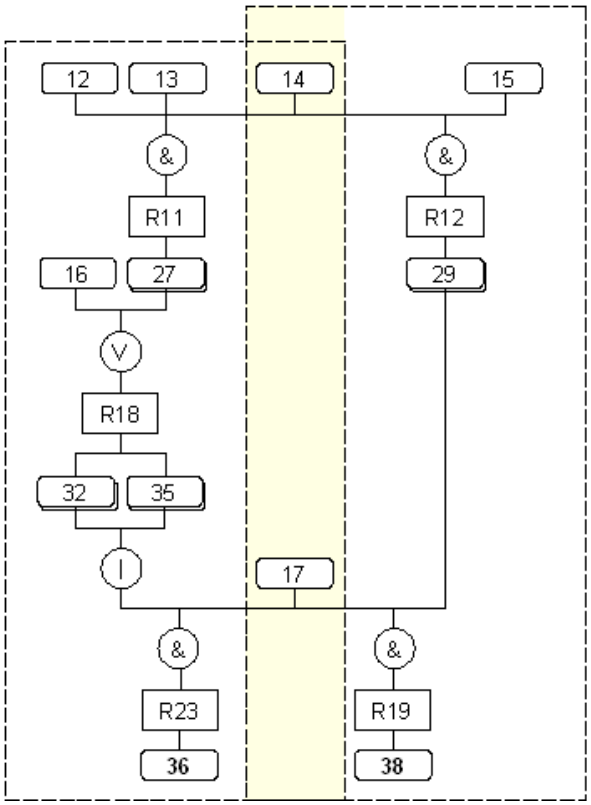

Figura 5.2: Cálculo del coeficiente de mezcla.

Obteniendo las bases informativas del par de objetivos y la cardinalidad asociada a las mismas, se tiene:

$$QP_{36} = \{12, 13, 14, 16, 17\} \qquad card(QP_{36}) = 5$$
$$QP_{38} = \{14, 15, 17\} \qquad card(QP_{38}) = 3$$

Y el conjunto $QP_{36} \cap QP_{38} = \{14, 17\}$, por tanto la cardinalidad de la intersección es 2. Con los valores anteriores se puede determinar el coeficiente de mezcla de estos objetivos como: $Cz = \frac{2\times 2}{3} - 1 = 0.3$, lo cual indica que existe coincidencia en las bases informativas de las proposiciones 36 y 38.

Los casos extremos para esta función se obtienen cuando el conjunto de intersección es vacío: $Cz = \frac{2\times 0}{x} - 1 = -1$ y cuando coincide con la cardinalidad de una de las bases informativas: $Cz = \frac{2\times x}{x} - 1 = 1$

La importancia del coeficiente de mezcla radica en la depuración de los objetivos de una BC, es decir una BC que posea objetivos totalmente independientes ($Cz = -1$), puede ser dividida para que los objetivos sean tratados como problemas no relacionados. De esta forma se delimita el alcance de las conclusiones de cada base resultante y se reduce la complejidad en su estructura.

5.3.1. Método para el cálculo del coeficiente de mezcla

En este caso, como el coeficiente está relacionado con todos los objetivos de la BC, el primer paso consiste en la extracción de las bases informativas; para ello se recorre la lista de proposiciones objetivos y se van almacenando las preguntas que se encuentren en sus líneas de razonamiento como se muestra en el algoritmo 8. Éste recibe la primera regla R donde la proposición objetivo es parte del sucedente y se comienza el análisis extrayendo el antecedente Ant de dicha regla. Luego se analizan las proposiciones que lo conforman, de ahí las que sean preguntas son almacenadas en Qp y las intermedios se convierten en sub-objetivos ejecutándose la función BaseInformativa de manera recursiva. Esto se efectúa mientras que se encuentren reglas donde p sea parte del sucedente (Próxima_regla_esSucedente(p, R)).

Algoritmo 8 Método para la extracción de la base informativa de un objetivo

Entrada: La primera regla donde aparece la proposición objetivo como sucedente, R.
Salida: La base informativa asociada al objetivo, Qp.

```
BaseInformativa(R, *&Qp)
*Ant ←Extrae_antecedente(R)
for cada proposición p en el antecedente Ant do
  TipoPro ← Determina_tipo_proposición(p)
  if TipoPro = 'Pregunta' then
    Qp ← Add(p)
  else
    R ← Saca_1raRegla_esSucedente(p)
    while R > 0 do
      BaseInformativa(R, Qp)
      R ← Proxima_regla_esSucedente(p, R)
```

Cuando se tiene almacenadas las bases informativas de todos los objetivos se calcula la cardinalidad de las intersecciones entre pares de objetivos y la cardinalidad mínima de los conjuntos involucrados (Qpi y Qpj). Estos valores se van acumulando (sumatoria de todos los objetivos) para finalmente aplicar la fórmula que define al coeficiente.

5.4. Coeficiente de profundidad

Otro factor que también sirve para el análisis de las estructuras internas de las BC, lo constituye el estudio de las líneas de razonamiento asociadas a los objetivos.

En una BC, un hecho indeseable lo constituye la presencia de proposiciones preguntas directamente relacionadas con los objetivos, ya que esto indica ausencia de razonamiento según la arquitectura.

Sin embargo, en este hecho no se tiene en cuenta que también es necesario estudiar los tamaños de estas cadenas, puesto que no es lo mismo una BC que tenga solamente un nivel de intermedios a otra que tenga 4 ó 5 niveles. Esto puede verse como un índice que mide el nivel de razonamiento potencial de una BC.

Definición 26 *Se llama* Coeficiente de Profundidad para un objetivo P_j *(CpP_j) a la expresión de la forma:*

$$CpP_j = \frac{e^{\overline{PLP_j}} - 2}{e^{\overline{PLP_j}}}$$

Donde $\overline{PLP_j}$ representa la profundidad promedio de las líneas de razonamiento asociadas a P_j, es decir:
$\overline{PLP_j} = \frac{\sum_{i=1}^{NLP_j} PL_i}{NLP_j}$, siendo NLP_j el número de líneas de razonamiento asociadas a P_j.

En general, lo que se define es una transformación del promedio de profundidad al intervalo [-1,1) para tener un valor numérico que permita un análisis más objetivo.

Valores positivos de CpP_j indican una relación de profundidad apropiada, (aunque valores alrededor de cero ya expresan poca profundidad en la BC) mientras que valores negativos hablan sobre la existencia de una relación muy desfavorable en las líneas de razonamiento. Cuando esto ocurre se recomienda al ingeniero del conocimiento incorporar o emplear conocimiento intermedio almacenado en la BC que permita deducir las conclusiones sin realizar preguntas directas al usuario.

Ejemplo: Las siguientes reglas expresan las líneas de razonamiento para la proposición objetivo 46 y se muestran en la figura 5.3.

$$\begin{aligned}
R5 &: 5\ V\ 6 \Rightarrow 25(-1\ 1)\\
R6 &: 7\ V\ 8 \Rightarrow 26(-1\ 1)\\
R7 &: 9 \mid 10 \Rightarrow 26(-0.7\ 1)\\
R8 &: 11\ \&\ 25 \Rightarrow 43(-0.2\ 0.5)\\
R9 &: 12\ \&\ 26 \Rightarrow 44(-0.4\ 1)\\
R10 &: 43 \mid 44 \Rightarrow \mathbf{46}(-1\ 1)
\end{aligned}$$

Las líneas de razonamiento para la proposición 46 son:

$$\begin{aligned}
L_1P_{46} &= \{R10, R8\} & PL_1 &= 1\\
L_2P_{46} &= \{R10, R8, R5\} & PL_2 &= 2\\
L_3P_{46} &= \{R10, R9\} & PL_3 &= 1\\
L_4P_{46} &= \{R10, R9, R6\} & PL_4 &= 2\\
L_5P_{46} &= \{R10, R9, R7\} & PL_5 &= 2
\end{aligned}$$

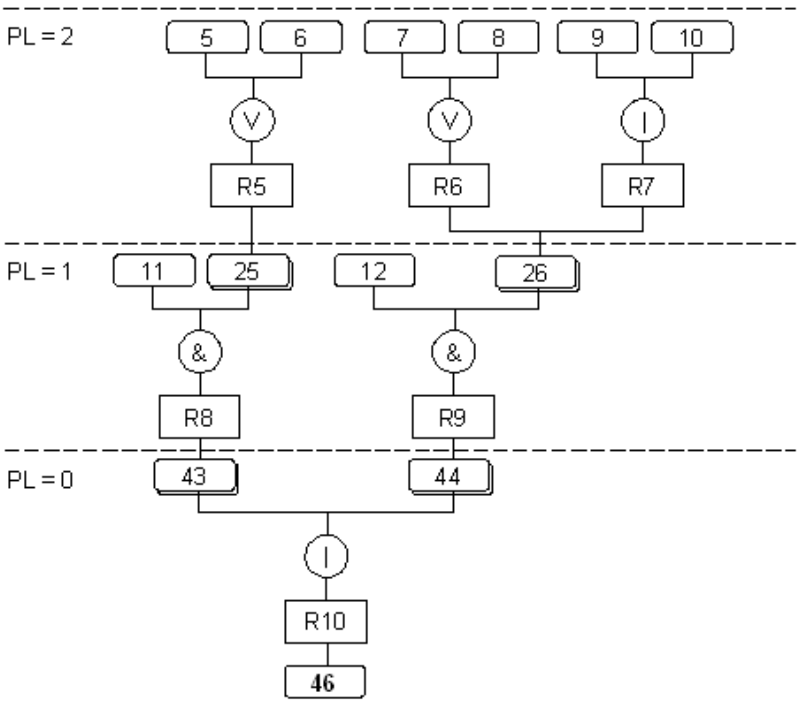

Figura 5.3: Cálculo del coeficiente de profundidad.

En L_5P_{46} se duplica la línea de razonamiento hasta la regla $R9$.

Y el promedio de dichas líneas es:

$$\begin{aligned} PLP_{46} &= (PL_1 + PL_2 + PL_3 + PL_4 + PL_5)/5 \\ &= (1 + 2 + 1 + 2 + 2)/5 \\ &= 1.6 \end{aligned}$$

Por tanto el coeficiente de profundidad se puede calcular como: $CpP_{46} = \frac{e^{1.6}-2}{e^{1.6}} = 0.6$, lo que demuestra que hay un nivel de profundidad promedio adecuado para la proposición que se analiza.

Los casos extremos se obtienen cuando la profundidad promedio de las líneas de razonamiento del objetivo es 0: $CpP_{46} = \frac{e^0-2}{e^0} = -1$ y cuando tiende a ∞:
$\lim_{\overline{PLP_j}\to\infty} \frac{e^{\overline{PLP_j}}-2}{e^{\overline{PLP_j}}} = 1$

Definición 27 *Se denomina* Coeficiente de Profundidad de la BC *(se denota Cp) a la siguiente expresión:*

$$Cp = \frac{e^{\overline{PL}} - 2}{e^{\overline{PL}}}$$

donde $\overline{PL}$ representa la profundidad promedio de las líneas de razonamiento de la BC completa, es decir:
$\overline{PL} = \frac{\sum_{i=1}^{NL} PL_i}{NL}$, siendo NL el número de líneas de razonamiento de la BC.

5.4.1. Método para el cálculo del coeficiente de profundidad

Aunque el coeficiente de profundidad puede estar relacionado todos los objetivos de la BC, el estudio del método que se sigue para su cálculo se ha acotado al análisis de un objetivo específico, puesto que a partir de éste se puede obtener la generalización al conjunto de objetivos.

El primer paso consiste en la extracción de las profundidades de todas las líneas de razonamiento asociadas al objetivo en cuestión. Para ello se utiliza el algoritmo 9 que al igual que en el algoritmo anterior, recibe la primera regla R donde el objetivo aparece como parte del sucedente.

Algoritmo 9 Método para la extracción de las profundidades de las líneas de razonamiento asociadas a un objetivo

---

Entrada: La primera regla donde aparece la proposición objetivo como sucedente, R.
Salida: Las profundidades de las líneas de razonamiento asociadas al objetivo, PLp.

ProfundidadesLR$(R, \& * PLp)$
$Prof \leftarrow PLp_N$
$*Ant \leftarrow$Extrae_antecedente(R)
for cada proposición p en el antecedente Ant **do**
  $TipoPro \leftarrow$ Determina_tipo_proposición(p)
  if $TipoPro =$ 'Intermedio' **then**
    $PLp_N \leftarrow PLp_N + 1$
    $R \leftarrow$ Saca_1raRegla_esSucedente(p)
    while $R > 0$ **do**
      ProfundidadesLR(R, PLp)
      $R \leftarrow$ Proxima_regla_esSucedente(p, R)
      $PLp \leftarrow$ Add$(Prof + 1)$

Se extrae el antecedente Ant y cuando se encuentran proposiciones p intermedios en él se incrementa en 1 el último elemento del arreglo de profundidades Plp y se comienza el análisis con esas proposiciones llamando de manera recursiva el método.

Esto se efectúa mientras que p se encuentre en el sucedente de alguna otra regla (Próxima_regla_esSucedente(p, R)) y cuando esto ocurre se duplica la línea de razonamiento recorrida hasta el momento para continuar con el cálculo de la nueva profundidad ($PLp \leftarrow \text{Add}(Prof + 1)$).

Con las profundidades del objetivo calculadas se determina el promedio y se aplica la fórmula que define el coeficiente.

5.5. Índice de razonamiento global

Otro elemento importante que ayuda a la valoración de estas estructuras, está relacionado con la cantidad de conocimientos asociado al proceso de razonamiento que deben ejecutar estos sistemas. Aquí este concepto se encuentra vinculado, fundamentalmente, a la cantidad de proposiciones intermedios y su relación con el resto de las proposiciones.

Definición 28 *Se llama* Índice de Razonamiento Global para la proposición objetivo P_j *($IRgP_j$) a la expresión:*

$$IRgP_j = \frac{2 \times card(IP_j)}{card(QP_j \cup IP_j)} - 1$$

donde IP_j denota el conjunto de proposiciones intermedios accesibles desde P_j y QP_j la base informativa asociada a P_j.

Como se puede observar este índice mide la proporción de proposiciones intermedio con relación a las preguntas. Esta relación puede ser usada como una idea general del volumen de razonamiento existente en una BC.

El índice $IRgP_j$ es un número que se encuentra en el intervalo [-1,1). Valores negativos indican una relación poco favorable entre intermedios y preguntas, en estos casos es necesario que el ingeniero del conocimiento disminuya la cantidad de preguntas asociadas al objetivo mediante la deducción de las mismas a través de intermedios. Por el contrario, valores positivos del coeficiente hablan sobre lo elaborado del razonamiento.

Ejemplo: En la figura 5.4 se presentan las cadenas de razonamiento de la proposición objetivo 53.

$$\begin{aligned}
R15: &\ 20\ V\ 21 \Rightarrow 33(-0.8\ 1)\\
R17: &\ 22\ \&\ 23 \Rightarrow 34(-1\ 0.5)\\
R28: &\ 33 \mid 34 \Rightarrow 41(-0.2\ 0.2)\\
R29: &\ 19\ \&\ 24 \Rightarrow 42(0\ \ -0.4)\\
R37: &\ 41\ V\ 42 \Rightarrow \mathbf{53}(-1\ 1)
\end{aligned}$$

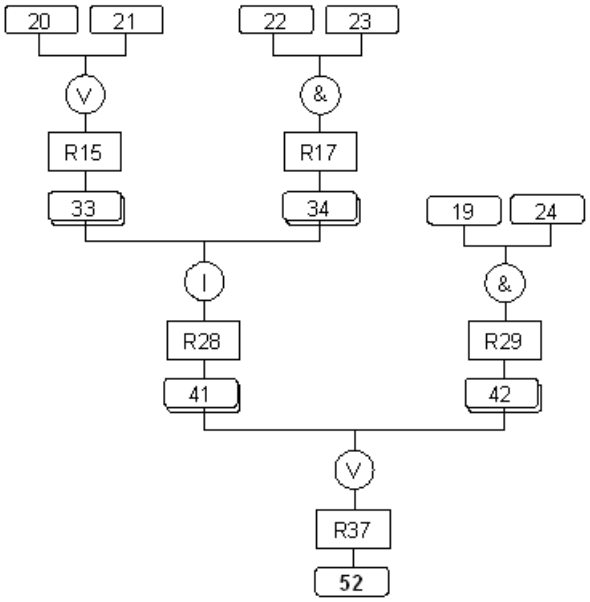

Figura 5.4: Cálculo del índice de razonamiento global.

Obteniendo el conjunto de proposiciones intermedios accesibles y la base informativa del objetivo 53, se tiene:

$$IP_{53} = \{33, 34, 41, 42\}, \qquad card(IP_{52}) = 4$$
$$QP_{53} = \{19, 20, 21, 22, 23, 24\}, \qquad card(QP_{52}) = 6$$

Y el conjunto $IP_{53} \cup QP_{53} = \{19, 20, 21, 22, 23, 24, 33, 34, 41, 42\}$, por tanto la cardinalidad de la unión es 10.

Una vez calculados todos los datos necesarios, se puede determinar el índice de razonamiento para el objetivo en cuestión como: $IRgP_{53} = \frac{2\times4}{10} - 1 = -0.2$, lo cual sugiere que existe una relación no favorable entre las proposiciones intermedios y preguntas accesibles desde 53.

El caso extremo de esta función se obtiene cuando el conjunto de intermedios es vacío: $IRgP_j = \frac{2\times0}{x} - 1 = -1$. Una característica interesante es que $IRgP_j$ nunca puede tomar valor 1, puesto que la base informativa asociada a una proposición no puede ser nula porque entonces el proceso de evaluación de ésta jamás terminaría.

Definición 29 *Se llama* Índice de Razonamiento Global de la BC *y se denota* IRg*, a la siguiente expresión:*

$$IRg = \sum_{j=1}^{No} IRgP_j = IRgP_1 \oplus_c IRgP_2 \oplus_c ... \oplus_c IRgP_{No}$$

donde la operación $\oplus_c$, si X y Y denotan dos valores cualesquiera de $IRgP_j$, se obtiene como sigue:

$$X \oplus_c Y = \begin{cases} -1 & si\ X = Y = -1 \\ \frac{X+Y}{1+X\times Y} & si\ X, Y \neq -1 \\ \frac{Z-0.999}{1-0.999\times Z} & si\ (X = -1\ y\ Y \neq -1) \quad o \quad (Y = -1\ y\ X \neq -1) \\ & con\ Z = \begin{cases} X & si\ Y = -1 \\ Y & si\ X = -1 \end{cases} \end{cases}$$

5.5.1. Método para el cálculo del índice de razonamiento global

Como se definió, existen dos modos de aplicación de este índice: a una proposición objetivo o a todos los objetivos de la BC. Para determinar el primero se utiliza algoritmo 10 que es muy parecido al 8.

Algoritmo 10 Método para la extracción de la base informativa y las preguntas intermedios accesibles desde un objetivo

Entrada: La primera regla donde aparece la proposición objetivo como sucedente, R.
Salida: La cardinalidad de la base informativa, $cardQp$ y del conjunto de proposiciones intermedios accesibles desde el objetivo, $cardIp$.

```
LineaRazonamiento(R, &cardQp, &cardIp)
*Ant ←Extrae_antecedente(R)
for cada proposición p en el antecedente Ant do
  TipoPro ← Determina_tipo_proposición(p)
  if TipoPro = 'Pregunta' then
    cardQp ← cardQp + 1
  else
    cardIp ← cardIp + 1
    R ← Saca_1raRegla_esSucedente(p)
    while R > 0 do
      LineaRazonamiento(R, cardQp, cardIp)
      R ← Proxima_regla_esSucedente(p, R)
```

La diferencia radica en que las proposiciones preguntas no son almacenadas sino contabilizadas al igual que las intermedios. Con estos valores se aplica la fórmula que define el índice para ser calculado.

Para obtener el índice global sobre todos los objetivos de la BC, se recorre la lista de los mismos y se aplica la función OPlus ($\oplus_c$), como fue especificada en la definición 29.

5.6. Índice de razonamiento directo

Existe una segunda forma de enfocar el análisis anterior. La idea consiste en considerar solamente aquellas proposiciones que son evidencias directas, es decir, que se encuentran como parte de los antecedentes de las reglas donde los objetivos son sucedentes, y ejecutar los cálculos a partir de ahí.

Es conocido que uno de los aspectos a tomar en cuenta durante el desarrollo de una BC, lo constituye la necesidad de evitar que las preguntas constituyan evidencias directas para evaluar los objetivos, puesto que ello indica ausencia de razonamiento.

La siguiente definición permite calcular un índice que brinda información en este sentido.

Definición 30 *Se llama* Índice de Razonamiento Directo para la proposición objetivo P_j *($IRdP_j$) a la expresión:*

$$IRdP_j = 1 - \frac{2 \times card(QdP_j)}{card(Qdp_j \cup IdP_j)}$$

donde IdP_j denota el conjunto de proposiciones intermedio que son evidencias directas de P_j
QdP_j el conjunto de proposiciones preguntas que son evidencias directas de P_j.

Como en el caso anterior, $IRdP_j$ es un número del intervalo [-1,1] y valores negativos indican una relación poco favorable entre evidencias intermedios y preguntas, mientras que valores positivos por el contrario hablan sobre lo elaborado del razonamiento.

Ejemplo: La proposición objetivo 45 se obtiene por medio de la siguiente regla, representada en la figura 5.5.

$$R31: \ 18 \ \& \ (33 \ V \ 34) \Rightarrow \mathbf{45}(-1 \ 1)$$

Los conjuntos de evidencias directas para 45 son:

$$QdP_{45} = \{18\}, \qquad card(QdP_{45}) = 1$$
$$IdP_{45} = \{41, 42\}, \qquad card(IdP_{45}) = 2$$

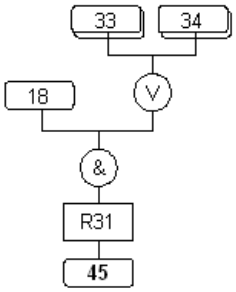

Figura 5.5: Cálculo del índice de razonamiento directo.

Y el conjunto $QdP_{45} \cup IdP_{45} = \{18, 41, 42\}$, por tanto la cardinalidad de la unión es 3 y el índice puede ser calculado como: $IRdP_{45} = 1 - \frac{2\times1}{3} = 0.3$, lo cual indica que hay una relación apropiada entre las preguntas e intermedios que son evidencias directas de 45.

Los valores extremos para este coeficiente se presentan para el caso en que el conjunto de intermedios es vacío: $IRdP_j = 1 - \frac{2\times x}{x} = -1$ y cuando el conjunto de preguntas es vacío: $IRdP_{45} = 1 - \frac{2\times0}{x} = 1$. De hecho se considera este último caso como ideal, pues ninguna pregunta es evidencia directa del objetivo.

Definición 31 *Se denomina* Índice de Razonamiento Directo de la BC *y se denota* IRd*, a la siguiente expresión:*

$$IRd = \sum_{j=1}^{No} IRdP_j = IRdP_1 \oplus_c IRdP_2 \oplus_c ... \oplus_c IRdP_{No}$$

donde la operación $\oplus_c$, si X y Y denotan dos valores cualesquiera de $IRdP_j$, se obtiene como sigue:

$$X \oplus_c Y = \begin{cases} -1 & si\ X = Y = -1 \\ 1 & si\ X = Y = 1 \\ \frac{X+Y}{1+X\times Y} & si\ X, Y \notin \{-1, 1\} \\ \frac{Z-0.999}{1-0.999\times Z} & si\ (X = -1\ y\ Y \neq -1) \quad o \quad (Y = -1\ y\ X \neq -1) \\ & con\ Z = \begin{cases} X & si\ Y = -1 \\ Y & si\ X = -1 \end{cases} \\ \frac{Z+0.999}{1+0.999\times Z} & si\ (X = 1\ y\ Y \neq 1) \quad o \quad (Y = 1\ y\ X \neq 1) \\ & con\ Z = \begin{cases} X & si\ Y = 1 \\ Y & si\ X = 1 \end{cases} \end{cases}$$

Como se puede observar se está definiendo un índice para la BC completa a partir de los valores correspondientes a cada objetivo.

5.6.1. Método para el cálculo del índice de razonamiento directo

Para el cálculo del índice de razonamiento directo asociado a un objetivo inicialmente se extraen las cardinalidades de las evidencias directas preguntas ($cardQdp$) e intermedios ($cardIdp$) de dicho objetivo empleando el algoritmo 11.

Algoritmo 11 Método para la extracción de las evidencias directas asociadas a un objetivo

---

Entrada: La regla donde aparece la proposición objetivo como sucedente, R.
Salida: Las cardinalidades de los conjuntos de evidencias directas asociadas al objetivo, $cardQdp, cardIdp$.

EvidenciasDirectas($R, \&cardQdp, \&cardIdp$**)**
$*Ant \leftarrow$Extrae_antecedente(R)
for cada proposición p en el antecedente Ant **do**
  $TipoPro \leftarrow$ Determina_tipo_proposición(p)
  if $TipoPro =$ 'Pregunta' **then**
    $cardQdp \leftarrow cardQdp + 1$
  else
    $cardIdp \leftarrow cardIdp + 1$

Con estos valores se aplica la fórmula que define al índice que se analiza. Para todos los objetivos de la BC, se recorre la lista de los mismos y se aplica la función OPlus ($\oplus_c$), como fue especificada en la definición 31.

5.7. Estudio empírico

En esta sección se analizan los resultados de aplicar los algoritmos descritos en los epígrafes anteriores al mismo conjunto de SBC, tomado para ejemplificar el comportamiento del módulo de verificación en el capítulo anterior[1]. En el cuadro 5.1 se incluye, además de los totales de reglas y proposiciones, el número de objetivos de cada BC.

[1]Véase descripción de los SBC en 4.8

BC	# Reglas	# Proposiciones	# Objetivos
TRANSFOR	69	93	6
ESFEDA	110	325	20
POZOS	116	85	7
EPILEP	943	862	32

Cuadro 5.1: Características de las BC

Después de ejecutar el módulo de análisis de coeficientes para las BC de los cuatro sistemas, se han recogido las cantidades de ciclos de reloj obtenidos corriendo la herramienta en una Pentium 4 a 2.66 GHz en el cuadro 5.2.

Coeficiente	TRANSFOR	ESFEDA	POZOS	EPILEP
Mezcla	0	10	0	587
Profundidad	0	10	0	603
Razonamiento global	0	10	0	595
Razonamiento directo	0	0	0	0

Cuadro 5.2: Ciclos de reloj en la ejecución de los métodos para el cálculo de los coeficientes

Como se puede apreciar, el número de ciclos depende de la cantidad de objetivos de la BC, puesto que ello tiene que ver directamente con el número de líneas de razonamiento asociadas a los mismos. Las longitudes de estas cadenas estriban en las características de la representación del conocimiento según el sistema.

Sólo en los casos de ESFEDA y EPILEP con 20 y 32 objetivos respectivamente se obtienen cantidades mayores que 0 y la diferencia entre ellos radica en el promedio del número de literales en los antecedentes de las reglas en las que se encuentran los objetivos formando parte de sus sucedentes.

En el índice de razonamiento directo, como se hace un análisis lineal de cada regla, no se alcanzan a distinguir aumentos en el tiempo, independientemente del número de objetivos de la base.

El cuadro 5.3 muestra el resultado de los cálculos de coeficientes realizados por VBCH. En todos los casos se realizó el análisis para toda la BC.

Coeficiente	TRANSFOR	ESFEDA	POZOS	EPILEP
Mezcla	0.59	-0.58	0.69	0.45
Profundidad	0.71	0.90	-0.68	0.84
Razonamiento global	-1.00	-1.00	-1.00	-1.00
Razonamiento directo	1.00	1.00	-1.00	-0.61

Cuadro 5.3: Valores de los coeficientes por caso de estudio

Los valores obtenidos del coeficiente de mezcla indican que en la mayoría de las bases existen coincidencias en las bases informativas asociadas a los objetivos. En el caso de ESFEDA habría que analizar la independencia entre las conclusiones que brinda el sistema y si es posible, dividir la BC puesto que no existe relación entre los objetivos de la misma.

En cuanto a los coeficientes de profundidad, casi todos los valores indican buen nivel de razonamiento, sólo en POZOS esta medida al ser negativa muestra la presencia de proposiciones preguntas directamente relacionadas con los objetivos, lo que sugiere ausencia de razonamiento según la arquitectura.

Los índices de razonamiento global en todas las BC revelan que la proporción entre los tipos de proposiciones involucradas en la evaluación de los objetivos no es favorable, ya que existen mucho más preguntas que intermedios accesibles desde los mismos.

Para el índice de razonamiento directo se obtiene el caso ideal en las bases TRANSFOR y ESFEDA, en el que no existen proposiciones preguntas como evidencias directas de los objetivos. En POZOS y EPILEP ocurre todo lo contrario, ya que hay muchas preguntas directamente relacionadas con la evaluación de sus objetivos, lo que indica ausencia de razonamiento.

5.8. Conclusiones

En este capítulo se han propuesto algunos coeficientes que sirven como medidas cualitativas para evaluar el conocimiento representado mediante las reglas de producción. Para cada uno de ellos se ha presentado un ejemplo que describe la forma en que se determina y el algoritmo que se implementó para calcularlo.

El resultado del cálculo y análisis de los coeficientes no quiere decir que la BC esté mal o bien, simplemente se obtiene una información que debe hacer reflexionar sobre la estructura que se ha construido. No se debe olvidar que en muchos problemas puede ocurrir, que el nivel de los conocimientos existentes no permita la elaboración de un sistema con grandes procesos de razonamiento, pero también que la detección y representación de estos, es difícil y constituye la base fundamental del éxito.

Capítulo 6

Sistema VBCH

El sistema de análisis y verificación de bases de conocimiento, VBCH ha sido concebido como la integración, en una herramienta, de tres módulos principales. Cada módulo cumple una función específica dentro del sistema, aunque se establecen relaciones entre ellos para lograr el desempeño exitoso de sus tareas. Se comienza estudiando entonces, la dependencia y la forma en que deben ser acoplados dichos módulos.

A cada módulo se dedica una sección del capítulo, donde son analizados a detalle partiendo de la función que realizan y sus diagramas de contextos. Se presentan además las entradas y salidas de cada etapa y los algoritmos que emplean para llevar a cabo la función correspondiente. Finalmente en esta etapa de diseño se muestra la interfaz del sistema y se presentarán ejemplos de aplicación de cada módulo con una BC que ha sido seleccionada como caso de estudio.

Por último, se especificará la forma en que han sido implementados estos módulos mediante sus estructuras de clases y la integración con las proporcionadas por el lenguaje HAries, según el paradigma orientado a objetos que brinda el lenguaje de programación empleado para su construcción, Visual C++.

6.1. Organización general del sistema

Como se ha mencionado el sistema VBCH está dividido en tres módulos principales:

1. Extracción de parámetros,
2. Verificación y
3. Análisis de coeficientes.

Cada uno de ellos puede ser utilizado de manera independiente por medio de una interfaz de usuario, en la que se especifica el módulo a acceder.

El módulo de extracción de parámetros tiene dos funciones primordiales: brindar un reporte sobre la composición de la BC y servir como intermediario entre el resto de los módulos y las estructuras de representación almacenadas. En la primera se encarga de proporcionar información relativa a cada estructura o a las relaciones que se establecen entre éstas. Cuando se utiliza como auxiliar del módulo de verificación se encarga de crear el meta-modelo del cual se obtendrá la información relevante y necesaria para hacer la detección de anomalías. Con el módulo de análisis de coeficientes se obtiene componentes específicos de las bases de proposiciones y reglas.

El módulo de verificación se encarga de la localización y descripción de las anomalías lógicas, semánticas y estructurales de la BC. Para ello utiliza los mecanismos de detección que se ocupan de analizar cada estructura haciendo un recorrido por el árbol de inferencias correspondiente.

El módulo de análisis de coeficientes es un estudio adicional sobre las bases descritas con el lenguaje HAries y que permite obtener información cuantitativa sobre las estructuras internas de representación empleadas.

El diagrama de contexto del sistema, en la figura 6.1, muestra los tres módulos referidos y las relaciones entre éstos.

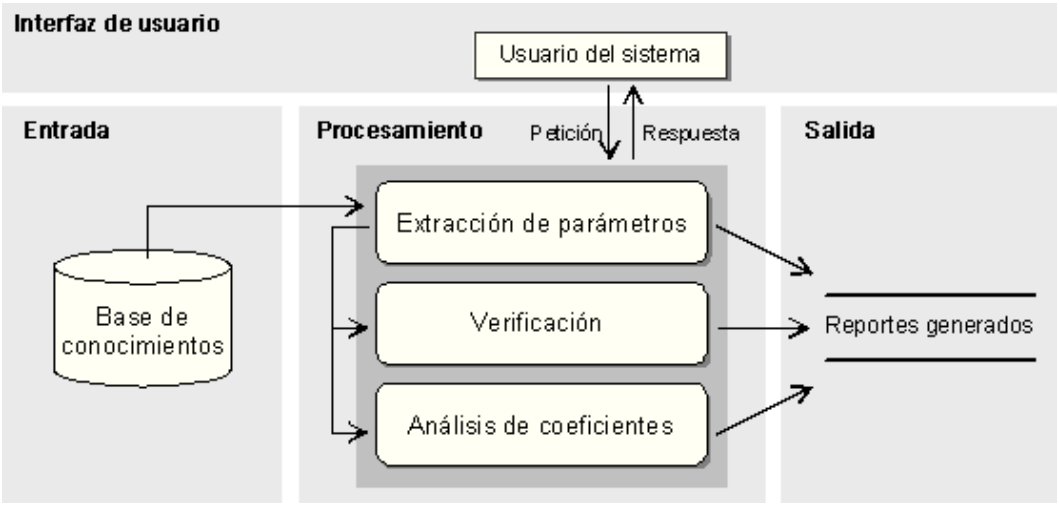

Figura 6.1: Diagrama de contexto de VBCH.

A continuación se discutirá el diseño e implemetación de cada módulo utilizando como caso de estudio la BC de un sistema para el pronóstico de la inyección de tenso - activos en pozos de petróleo: POZOS[1].

[1]Sistema descrito en 4.8

6.2. Módulo de extracción de parámetros

El módulo de extracción brinda al programador información sobre la composición de las bases de conocimiento, además de servir como intermediario para establecer la comunicación entre éstas y el resto de los módulos. Consiste en la construcción de tablas de parámetros estadísticos simples y gráficos.

Durante la consulta de una BC el ingeniero del conocimiento puede tener acceso a los parámetros generales del sistema, el estado de los archivos e información específica de las estructuras de representación (proposición y reglas de producción) basada en el conteo sus elementos o características y el cálculo de porcentajes, con lo cual se puede inferir la complejidad de la base. Además este módulo provee informes de la forma en que se establecen todas relaciones posibles entre las estructuras mencionadas.

En la figura 6.2 está representado el diagrama de contexto para este módulo del sistema.

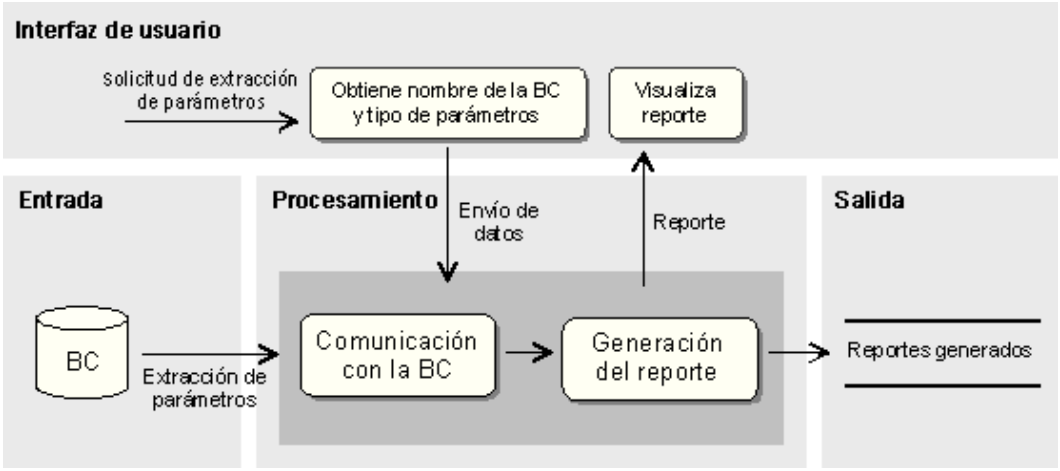

Figura 6.2: Diagrama de contexto del módulo de extracción de parámetros.

6.2.1. Especificación del módulo de extracción

El módulo de extracción está dividido en tres partes fundamentales, de acuerdo al tipo de parámetros que son reportados:

- *Parámetros generales*
- *Parámetros específicos*
- *Relaciones entre estructuras*

Dentro de los parámetros generales se presenta información de la cantidad de elementos de cada estructura y el estado de los archivos:

- Número de proposiciones que conforman la BC
- Total de reglas de producción generalizadas
- Número de conclusiones del sistema
- Preguntas potenciales
- Nombre de los archivos (texto o binarios) de la BC

Los parámetros específicos dependen del tipo de estructura de representación que se seleccione e indican cantidades y porcentajes según el total. El cuadro 6.1 presenta un resumen de las características que son reportadas para las estructuras principales de representación: proposiciones y reglas de producción.

Estructura de representación	Parámetros que se reportan
Proposiciones	- Cantidad de proposiciones - Atributos naturales (preguntas, intermedios, objetivos) - Tipo de dinamismo (estáticas, dinámicas) - Acciones antes y después de evaluar (mensajes, hipertextos, imágenes, videos, sonidos, programas externos) - Relaciones (proposiciones asociadas, contextuales, de evaluación alternativa)
Reglas de producción	- Cantidad de reglas - Tipo de antecedente - Cantidad de sucedentes - Pesos definidos

Cuadro 6.1: Características que se reportan de cada estructura

Las relaciones también están divididas según la estructura de representación de la cual parte la asociación. En esta parte no se enumeran cantidades, sino el número o identificador de las estructuras que establecen la correspondencia. Las relaciones que puede mostrar el sistema están recogidas en el cuadro 6.2.

Estructura de representación	Relaciones que se reportan
Proposiciones	- Proposición y lista de proposiciones asociadas - Proposición y lista de proposiciones que aparecen como parte de la condición en la relación contextual - Proposición y lista de proposiciones que aparecen en la relación de evaluación alternativa
Reglas de producción	- Regla de producción generalizada y lista de proposiciones que forman parte del antecedente - Regla de producción generalizada y lista de proposiciones que forman parte del sucedente

Cuadro 6.2: Relaciones entre las estructuras de representación

El algoritmo principal del módulo de extracción recibe como entrada: el nombre y ubicación de la BC, la especificación de los parámetros a extraer de las estructuras de representación del conocimiento y brinda como salida: un reporte generado en dependencia de los parámetros que hayan sido seleccionados.

Los siguientes son los pasos que se siguen en este módulo para extraer parámetros de una BC construida con HAries:

1. Recibe la ubicación y el nombre de la base.
2. Se selecciona el tipo de parámetros (generales, particulares, relaciones) a extraer y la categoría correspondiente.
3. Carga la BC a analizar en memoria, mediante los archivos de texto o binarios correspondientes.
4. Interpreta las estructuras almacenadas en la base de conocimientos.
5. Genera reporte con los parámetros adecuados a la consulta actual.
6. Visualiza el reporte elaborado.

6.2.2. Diseño de la interfaz del módulo de extracción

La pantalla principal del módulo de extracción se muestra en la figura 6.3. La barra de menús está dividida en ocho grupos de operaciones:

- *Base de Conocimientos*: Contiene las opciones para buscar, abrir la base de conocimientos más reciente o salir de ejecución.
- *Generales*: Sus opciones brindan información sobre la composición de la BC o el estado de los archivos que la conforman.
- *Específicas*: Despliega las dos estructuras de representación (proposiciones y reglas de producción) para en función de la selección generar el reporte correspondiente.
- *Relaciones*: Indica los pares de estructuras entre las que se pueden establecer asociaciones.
- *Impresión*: Sus opciones permiten la configuración de las impresoras instaladas, la presentación preliminar y la impresión del reporte generado.
- *Ver*: Controla la visualización de la barra de herramientas y de estado.
- *Ventana*: Sus funciones permiten la manipulación de todos los reportes elaborados durante una ejecución del sistema.

- *Ayuda*: Proporciona acceso a un hipertexto de ayuda e información de la versión actual del sistema.

La barra de herramientas duplica las órdenes del menú *Generales, Específicas* y *Relaciones* con el objetivo de facilitar o acelerar la acción sobre cada elemento.

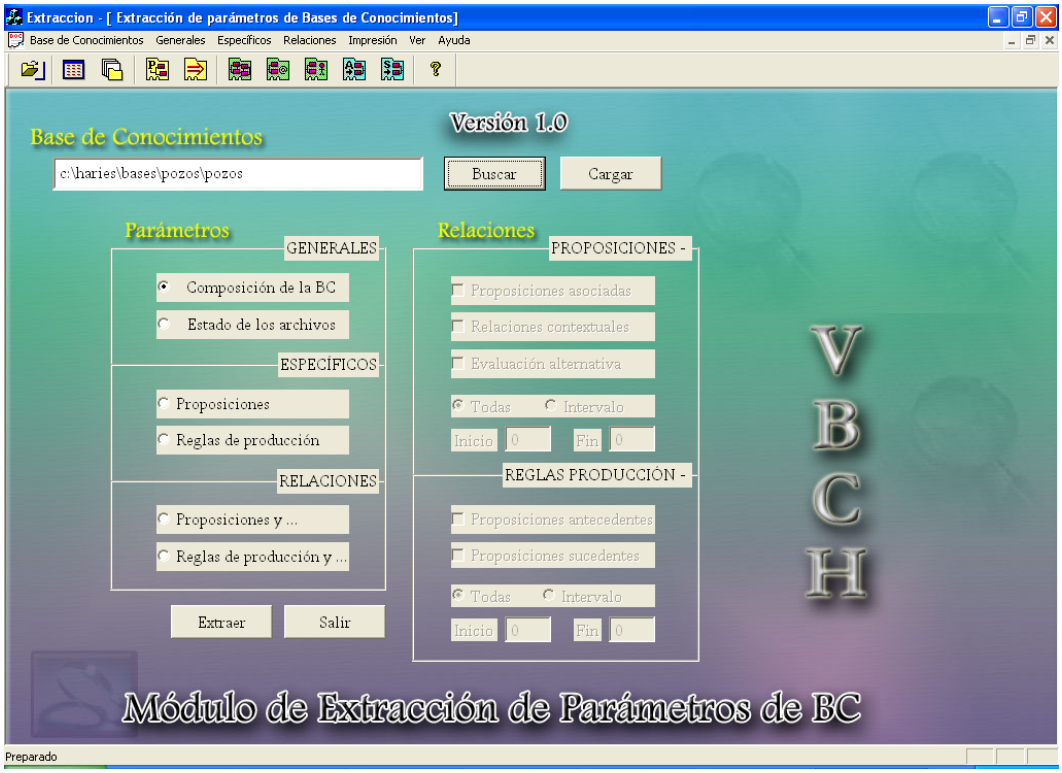

Figura 6.3: Interfaz de usuario del módulo de extracción de parámetros.

En el cuadro de texto el usuario introduce la ruta y el nombre de la BC de la cual se va realizar la extracción. Seguidamente, elige el tipo de parámetro y se presiona el botón *Extraer* que realiza la comunicación con la BC y obtiene los elementos indicados para elaborar el reporte.

En el caso de las relaciones entre estructuras hay que seleccionar además el par que compone a la relación y el intervalo de extracción. El análisis puede incluir todos los elementos de la estructura que define la relación (proposiciones o reglas) o un intervalo específico para un conjunto determinado.

6.2.3. Ejemplo de aplicación del módulo de extracción

En la figura 6.4 se muestran los datos generales de POZOS[1] en el reporte generado por el módulo de extracción, tras haber seleccionado como parámetro la composición de la BC.

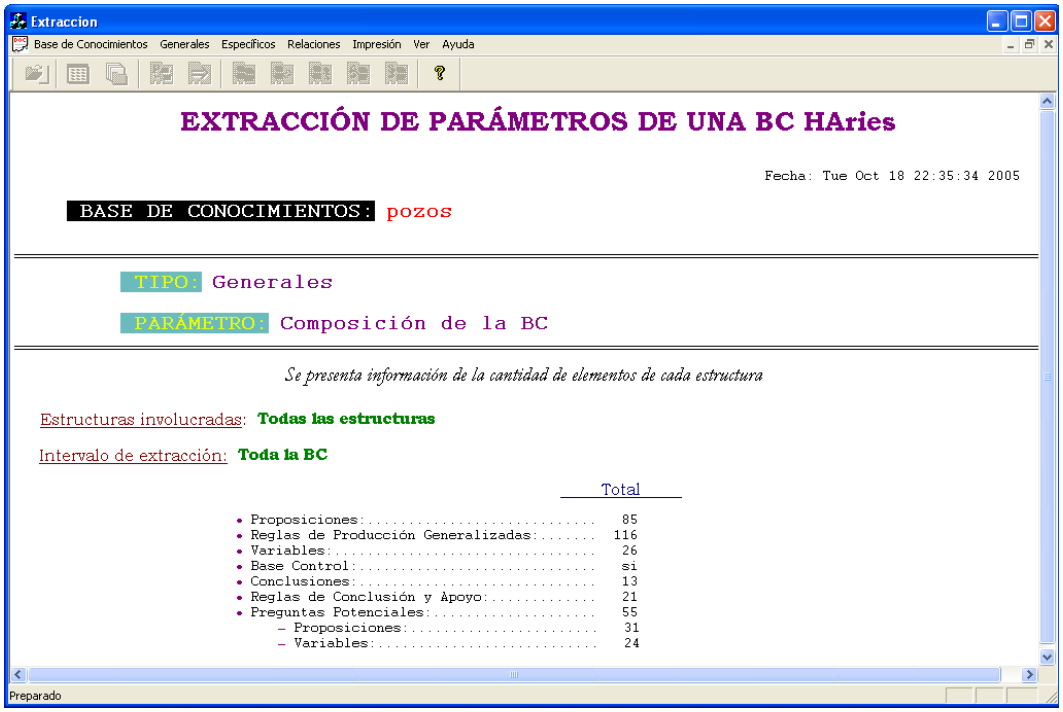

Figura 6.4: Ejemplo de un reporte generado en el módulo de extracción.

Como se puede observar la BC posee 76 proposiciones y 116 reglas de producción generalizadas, que son las estructuras involucradas en los siguientes módulos.

6.2.4. Estructura de clases del módulo de extracción

Para la implementación computacional del módulo de extracción se utilizó la jerarquía de clases de la figura 6.5.

A continuación se presenta una breve descripción de cada una de las clases empleadas y los métodos que implementa.

CExtraccionView

En esta clase se define la interfaz de usuario y su operación. Los métodos implementados se ocupan de la búsqueda o carga en memoria de la base de conocimientos

[1]BC del sistema descrito en 4.8

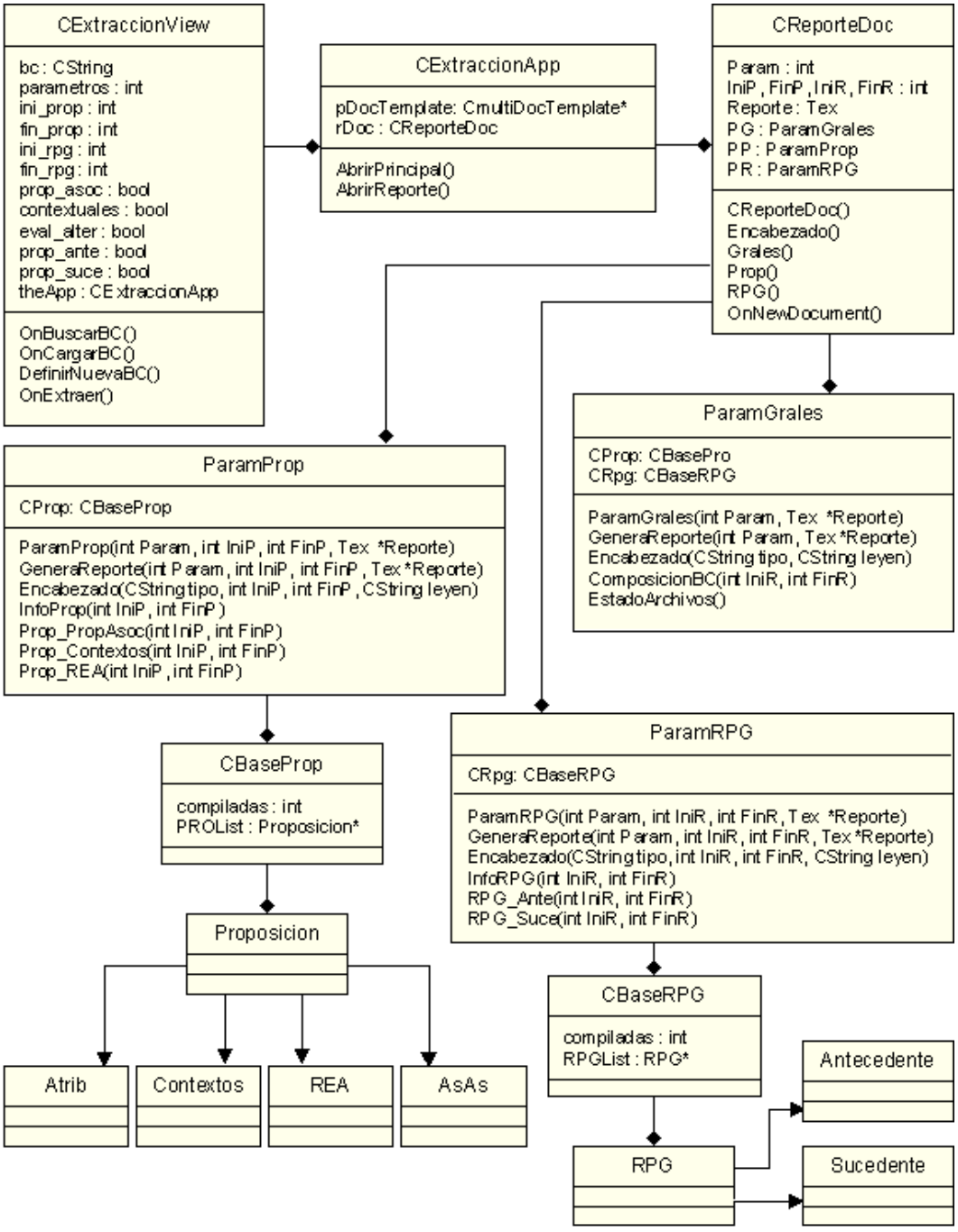

Figura 6.5: Clases para la extracción de información.

a analizar (*OnBuscarBC, OnCargarBC, DefinirNuevaBC*). La función *OnExtraer* emplea los atributos de la clase, inicializados mediante el diálogo de interacción, para generar el reporte y se ejecuta con la declaración del objeto *theApp* del tipo *CExtraccionApp* y la llamada a *AbrirReporte* de dicha clase.

CExtraccionApp

Se encarga de la manipulación de los documentos. El método *AbrirPrincipal* muestra la interfaz del módulo y *AbrirReporte* el documento que contendrá los parámetros solicitados. En este último se declara un objeto *rDoc* de la clase *CReporteDoc*.

CReporteDoc

Se emplea para la generación del reporte. En el constructor se llama a la función adecuada, según *Param*: - Si el valor de este atributo se encuentra en el rango que indica la extracción de parámetros generales se ejecuta el método *Grales* en el que se declara un objeto *ParamGrales* (PG), - si *Param* indica la extracción de parámetros de proposiciones se ejecuta el método *Prop* y se declara un objeto *ParamProp* (PP), - en otro caso (extracción de parámetros de reglas), se ejecuta el método *RPG* en el que se declara un objeto *ParamRPG*(PG).

Los atributos *IniP, FinP, IniR* y *FinR* indican el intervalo involucrado durante el proceso de extracción (poseen el mismo significado para el resto de las clases). La función *Encabezado* escribe dentro del documento *Reporte* el nombre de la BC y la fecha y hora actuales.

ParamGrales

Se utiliza para obtener información general de las estructuras y los archivos que conforman la BC. El método *GeneraReporte* va añadiendo en el documento de texto el resultado de las funciones *ComposicionBC* y *EstadoArchivos*, que extraen información sobre la estructura de la base y los archivos de texto y binarios en existencia.

La función *Encabezado* en esta y las próximas clases especifica el parámetro que se está reportando y una *leyenda* que describe brevemente en qué consiste.

ParamProp

Se ocupa para extraer información de la base de proposiciones. Nuevamente el método *GeneraReporte* añade en el documento la información que se adquiere según *Param*. La función *InfoProp* se utiliza para sacar los parámetros específicos de la proposición (señalados en tabla 6.1) en el intervalo definido por *IniP* - *FinP*. Por otra parte, *Prop_PropAsoc, Prop_Contextos* y *Prop_REA* se encargan de reportar las relaciones con otras proposiciones, contextuales y de evaluación alternativa respectivamente, para cada proposición en el intervalo.

ParamRPG

Obtiene información de la base de reglas de producción. La función *InfoRPG* se utiliza para sacar los parámetros específicos de las reglas en el intervalo *IniR - FinR*. Las relaciones entre los antecedentes y sucedentes de la reglas son reportadas por *RPG_Ante* y *RPG_Suce*.

En las tres clases anteriores se definen los objetos *CProp* y *CRpg* de las clases *CBasePro* y *CBaseRpg* definidas por el lenguaje. Éstas se encargan de la manipulación de las bases de proposiciones y de reglas de producción respectivamente, mediante una lista de la estructura correspondiente. Para todos los casos de extracción se realiza un recorrido de las listas para analizar cada estructura de manera individual. Cada elemento de dicha lista pertenece a la clase *Proposicion* o *RPG* según sea el caso. Los parámetros particulares en relación a *Proposicion* son definidos en sus clases bases (*Atrib, AsAs, Contextos* y *REA*), mientras que los de *RPG* son heredados de *Antecedente* y *Sucedente*.

6.3. Módulo de verificación

El módulo de verificación fue planeado para procurar algunas facilidades al ingeniero del conocimiento en el proceso de desarrollo de un SBC, específicamente de la construcción de la BC. Se encarga de la detección de las siguientes anomalías:

- *Semánticas*: Valores ilegales
- *Estructurales*: Redundancia y circularidad.
- *Lógicas*: Inconsistencias en reglas.

Estas anomalías fueron explicadas a detalle en el capítulo 4 y pueden provocar en el sistema diferentes consecuencias. Por ejemplo la presencia de duplicación de reglas en la BC degrada la ejecución del sistema y además puede originar problemas cuando se trate de extender, refinar o simplemente darle mantenimiento al sistema. Mientras que las secuencias circulares de reglas provocan ciclos infinitos, puesto que el sistema de inferencia no los reconoce en tiempo de ejecución.

La presencia de estos errores en bases descritas con el lenguaje HAries surgen a partir de la manera en que está concebido su sistema de adquisición, en el cual cada estructura es incorporada a la base de forma independiente. Siendo esto lo que provoca que en bases con un número considerable de elementos sea difícil detectar anomalías que involucren más de un objeto de la misma estructura. Por otra parte, resulta aun más complejo localizar las fallas que se producen a partir de las relaciones que se establecen entre estructuras diferentes.

Durante el curso de ejecución del módulo de verificación, se generan reportes en función de la solicitud que se realice. Estos reportes especifican de manera detallada los errores detectados y las estructuras involucradas en los mismos. En la figura 6.6 se muestra el diagrama de contexto para este módulo del sistema.

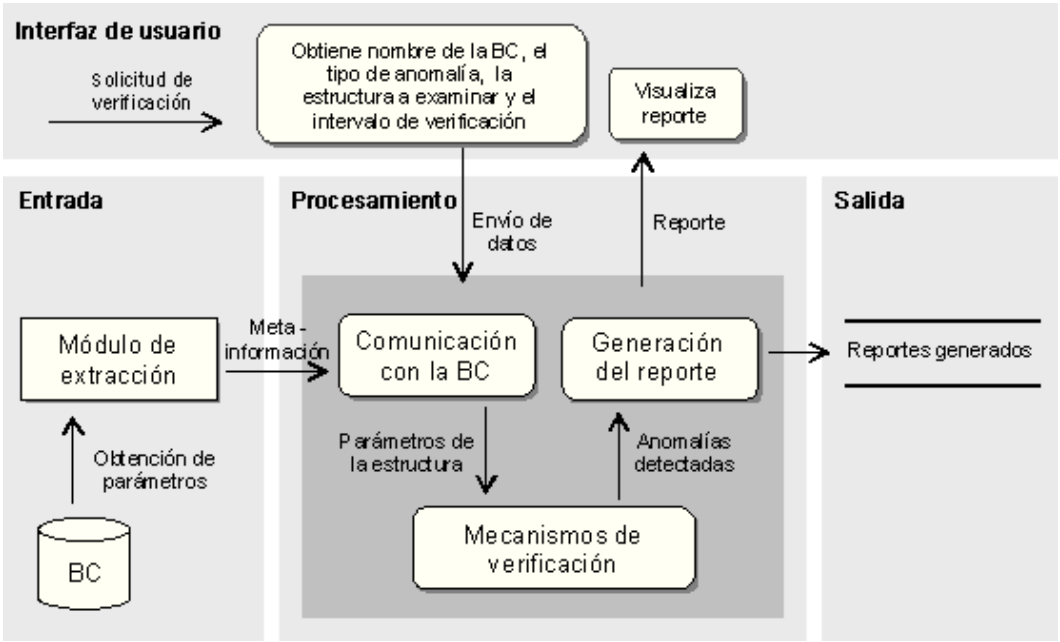

Figura 6.6: Diagrama de contexto del módulo de verificación.

6.3.1. Especificación del módulo de verificación

El módulo de verificación está dividido en dos partes fundamentales, la selección de la anomalía a tratar y la selección de la estructura e intervalo de verificación, para dicha estructura.

Anomalía	Proposiciones	Reglas
Valores ilegales	×	×
Inclusión		×
Duplicación		×
Ciclos locales	×	×
Ciclos particulares	×	×
Ciclos globales	×	×
Contradicción		×
Condiciones innecesarias		×

Cuadro 6.3: Anomalías que se reportan de cada estructura

El cuadro 6.3 es un resumen de los tipos de errores que se localizan en las estructuras proposición y regla de producción.

El algoritmo principal del módulo de verificación recibe como datos de entrada: el nombre y ubicación de la BC, la selección de la anomalía y estructura(s) a verificar, con el intervalo a considerar durante el proceso y brinda como salida: un reporte generado en dependencia de los errores detectados de la(s) estructura(s) seleccionada.

Los siguientes son los pasos que se siguen en este módulo para detectar las anomalías en una BC construida con HAries:

1. Recibe la ubicación y el nombre de la base.
2. Se seleccionan los elementos que formarán parte de la verificación (anomalía específica, estructura(s), intervalos a considerar)
3. Establece comunicación con la base por medio del módulo de extracción.
4. Recibe la meta-información, construida en la etapa de extracción de parámetros, en correspondencia con la elección de los elementos a considerar en el proceso de verificación.
5. Ejecuta los mecanismos de detección de anomalías apropiados.
6. Genera reporte con los problemas localizados.
7. Visualiza el reporte elaborado.

6.3.2. Diseño de la interfaz del módulo de verificación

La pantalla principal del módulo de verificación se muestra en la figura 6.7.

La barra de menús está dividida en seis grupos de operaciones:

- *Base de Conocimientos*: Contiene las opciones para buscar, abrir la base de conocimientos más reciente o salir de ejecución.
- *Estructura*: Se elige la estructura de representación (proposiciones o reglas de producción) para realizar la detección de todas las anomalías.
- *Anomalías*: Indica los problemas que detecta el sistema (valores ilegales, inclusión, duplicación, ciclos, contradicción y condiciones innecesarias) y ejecuta la verificación de las dos estructuras con el error señalado.
- *Ver*: Controla la visualización de la barra de herramientas y de estado.
- *Ventana*: Sus funciones permiten la manipulación de todos los reportes elaborados durante una ejecución del sistema.

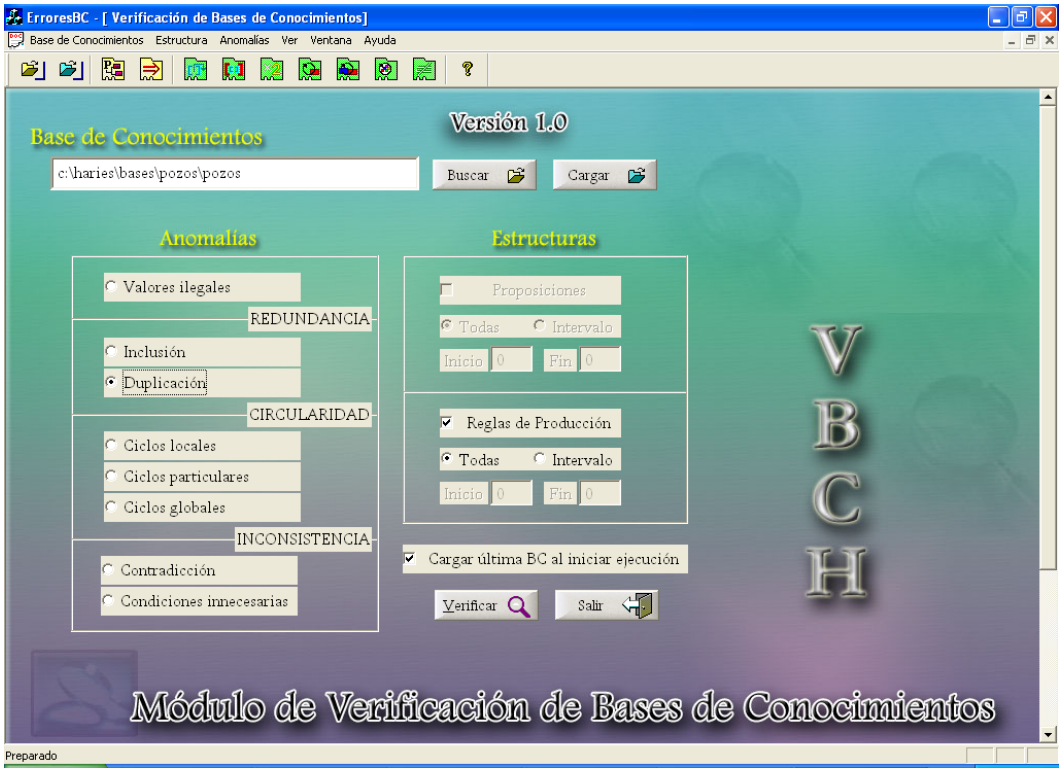

Figura 6.7: Interfaz de usuario del módulo de verificación.

- *Ayuda*: Proporciona acceso a un hipertexto de ayuda e información de la versión actual del sistema.

La barra de herramientas duplica las órdenes del menú, *Estructura* y *Anomalías* con el objetivo de facilitar o acelerar la acción sobre cada elemento.

En el cuadro de texto el usuario introduce la ruta y el nombre de la BC a ser verificada. Seguidamente, elige la anomalía y la estructura a analizar. El análisis puede incluir todos los elementos de la estructura seleccionada (base de proposiciones o de reglas) o un intervalo para la detección de los problemas en un conjunto determinado.

Una vez que los parámetros de verificación han sido seleccionados, se presiona el botón *Verificar* que ejecuta el método de detección correspondiente y despliega el reporte generado con las especificaciones de los errores descubiertos.

6.3.3. Ejemplo de aplicación del módulo de verificación

Utilizando la misma BC (POZOS) se ha ejecutado el método de verificación que detecta reglas duplicadas y el resultado para este análisis se muestra en la figura 6.8.

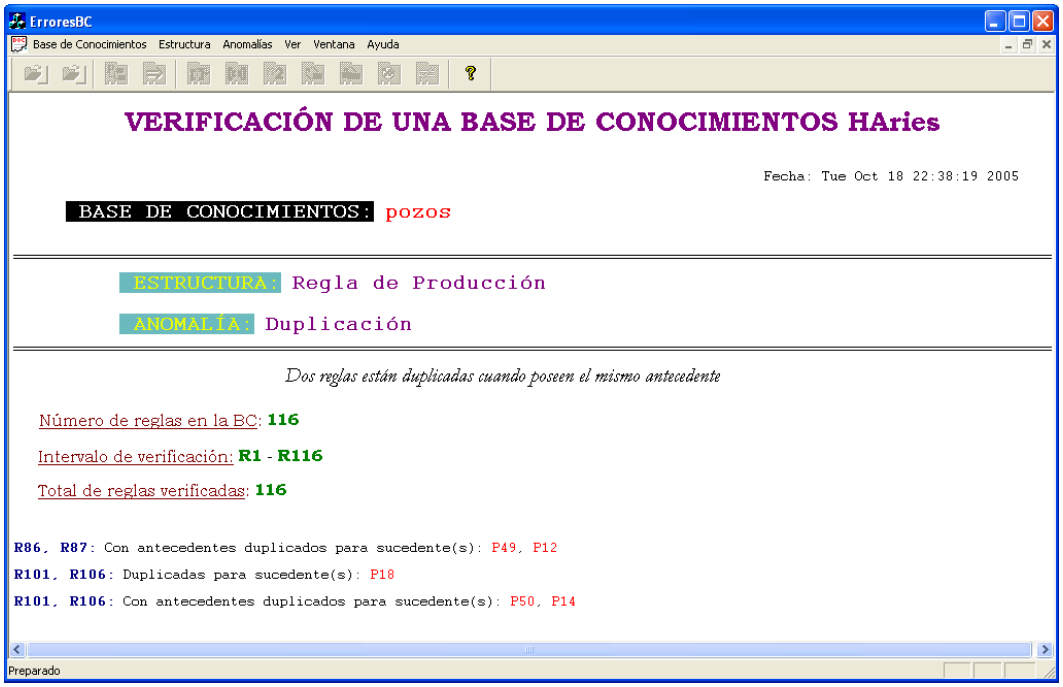

Figura 6.8: Ejemplo de un reporte generado en el módulo de verificación.

En el encabezado del reporte generado se incluye, además del nombre de la base y fecha del análisis, la estructura que se ha considerado en el proceso de detección, la anomalía tratada y el intervalo de verificación, que en este caso coincide con el total de reglas de la BC. En la parte de especificación para la base de POZOS se han detectado dos pares de reglas con problemas de duplicación: 86, 87 y 101, 106.

En el primer par se reportan antecedentes duplicados para las proposiciones sucedentes: $P49$ y $P12$. Para corroborar este resultado, se localizan las reglas señaladas en el sistema de adquisición de HAries:

$$R86: \ 8 \ \& \ -24 \ \& \ (-33 \ V \ -36 \ V \ -25 \ V \ -26) \Rightarrow 12(-0.5 \ 0.5);$$
$$R87: \ (-25 \ V \ -33 \ V \ -26 \ V \ -36) \ \& \ -24 \ \& \ 8 \Rightarrow 49(0 \ -0.8);$$

Y como se observa, aunque con diferente orden, los antecedentes de las reglas son iguales.

En el segundo caso se distinguen los tipos de duplicación de 101 y 106: para el sucedente $P18$ hay duplicación de reglas ($P18$ se encuentra en ambas) y para $50, P14$ hay duplicación del antecedente. Si extraemos las reglas se puede confirmar el resultado:

$$R101: \ (24 \mid 26) \ \& \ -32 \ \& \ (8 \mid -10 \mid -28) \Rightarrow 14(-0.25 \ 0.25), 18(0.33 \ 0);$$
$$R106: \ (-28 \mid 8 \mid -10) \ \& \ (26 \mid 24) \ \& \ -32 \Rightarrow 50(-0.25 \ 0.25), 18(0.33 \ 0);$$

6.3.4. Estructura de clases del módulo de verificación

La jerarquía de clases empleadas en este módulo se presenta en la figura 6.9. La manipulación del ambiente general se realiza de la misma forma que en el módulo de verificación.

A continuación se presenta una breve descripción de las clases empleadas para la detección de las anomalías. El resto de las clases son equivalentes a las definidas en el módulo de extracción.

AnomaliasProp

Se encarga de la manipulación de la base de proposiciones durante la verificación y de la generación del reporte correspondiente. El constructor recibe el tipo de anomalía a detectar *Anom*, el intervalo de verificación *IniP* - *FinP* (tienen el mismo significado en todas las funciones y clases del módulo) y el documento de texto donde se escribe el reporte *Reporte*. La función *ReportaAnomalia* hace la llamada al método correspondiente y el registro de los errores en el documento.

La función *Encabezado* se utiliza en esta y el resto de las clases para incluir en el reporte el *tipo* de anomalía, el intervalo de análisis y una *leyenda* que explica brevemente en qué consiste el problema.

Los métodos *ValoresIlegales, CiclosLocales* y *CiclosParticulares* se emplean para la detección de estas anomalías en la base de proposiciones y la elaboración de las descripciones que serán incluidas en el reporte. *ComparaLiterales* implementa el algoritmo 2 del capítulo 4 que realiza la inspección de las relaciones definidas en cada proposición P con el número de proposiciones de la BC *NProp*. En la localización de los ciclos se usa la función *PropCiclos* que implementa el algoritmo 7.

AnomaliasRPG

Se emplea para la detección de las anomalías en la base de reglas y la construcción del reporte para estas estructuras. El constructor y el método *ReportaAnomalia* en esta clase, tienen las mismas funciones que en *AnomaliasProp*.

El método *ValoresIlegales* realiza el recorrido por la base de reglas y utiliza *ComparaLiterales* que efectúa el algoritmo 1 para la búsqueda de proposiciones fuera de rango en el antecedente y sucedente de cada regla R. Además construye las explicaciones que serán incorporadas en el reporte.

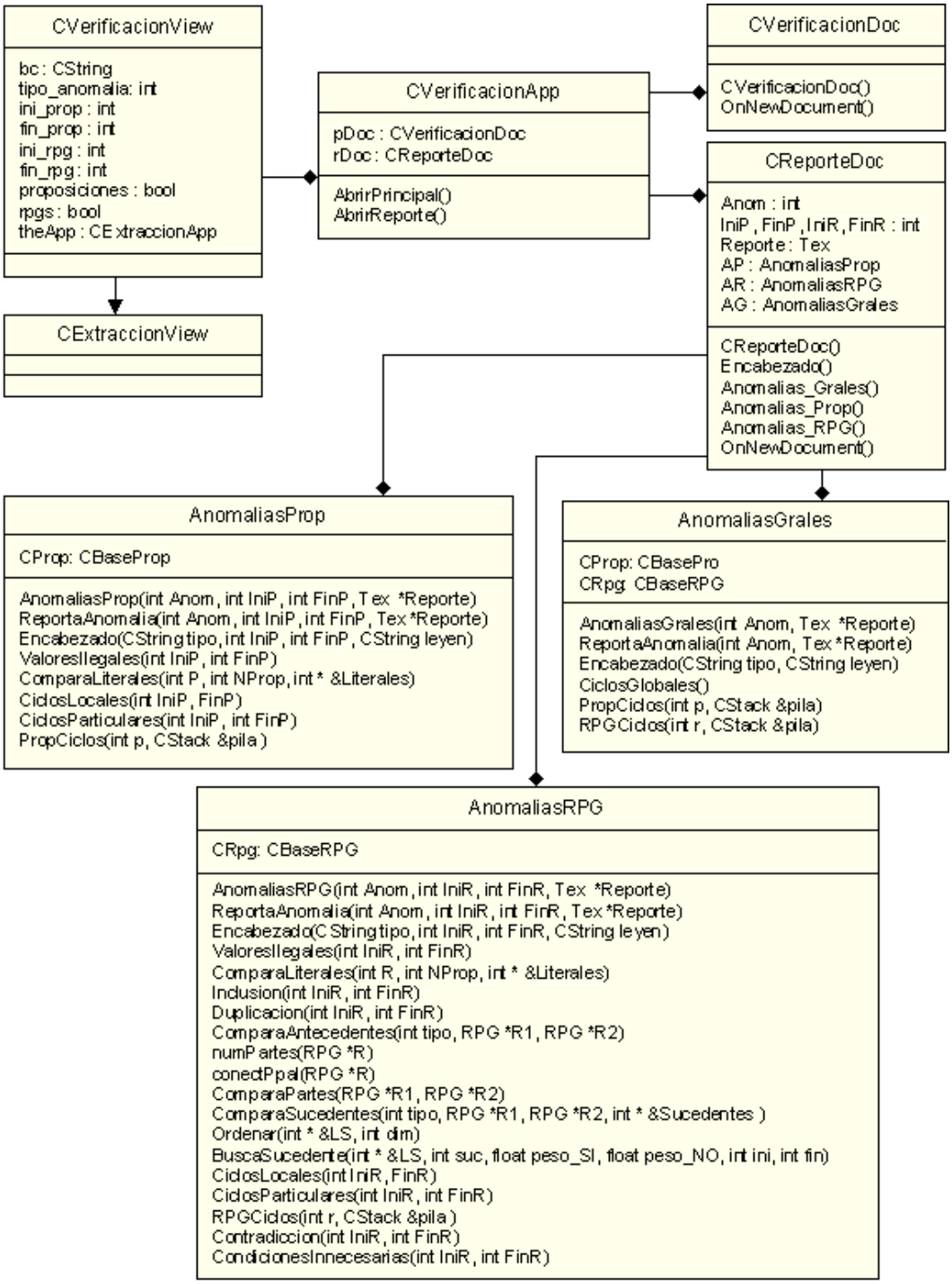

Figura 6.9: Clases para la detección de anomalías.

Las funciones *Inclusión* y *Duplicación* implementan el algoritmo 3, en el que se indica cuál de las anomalías se está tratando, conjuntamente estas funciones elaboran las descripciones con los resultados obtenidos del algoritmo y utilizan los métodos

ComparaAntecedentes y *ComparaSucedentes* durante el mismo.

Las funciones *numPartes, conectPpal* y *ComparaPartes* son ocupadas durante la comparación del antecedente de dos reglas y fueron descritas con la explicación del algoritmo 5. Del mismo modo *Ordenar* y *BuscaSucedente* se utilizan para la comparación del sucedente y se comentaron con el algoritmo 4.

Los ciclos en las reglas se reportan por medio de *CiclosLocales* y *CiclosParticulares*. Estos métodos usan *RPGCiclos* (implementa el algoritmo 6) para realizar la detección.

Por último, *Contradicción* y *CondicionesInnecesarias* también implementan el algoritmo 3, con los parámetros correspondientes e igualmente utilizan las funciones que en él se ocupan.

AnomaliasGrales

Se encarga de la detección e impresión de ciclos globales que involucran las dos estructuras. El método *CiclosGlobales* elabora las descripciones que serán incluidas en el reporte y dirige la búsqueda durante el proceso de detección de este error. Para la exploración de cada estructura particular utiliza las funciones *PropCiclos* y *RPGCiclos*.

6.4. Módulo de análisis de coeficientes

El módulo de análisis coeficientes se encarga del cálculo e interpretación de ciertas medidas que tienen que ver con la complejidad de la representación del conocimiento en la base de reglas.

Durante el proceso de estudio de una base, el ingeniero del conocimiento puede estar al tanto de la forma en que se encuentran asociadas las bases informativas del conjunto de proposiciones objetivos (coeficiente de mezcla) y así determinar el grado de solapamiento entre ellos.

Además el módulo permite examinar la relación entre las proposiciones preguntas e intermedios (índice de razonamiento global) y evidencias (índice de razonamiento directo) accesibles desde un objetivo, con el fin de obtener proporciones que permitan determinar el volumen de razonamiento de la BC. Por otra parte, se pueden evaluar las líneas de razonamiento de un objetivo determinando la cantidad de niveles que posee (coeficiente de profundidad).

En la figura 6.10 se muestra el diagrama de contexto para este módulo del sistema.

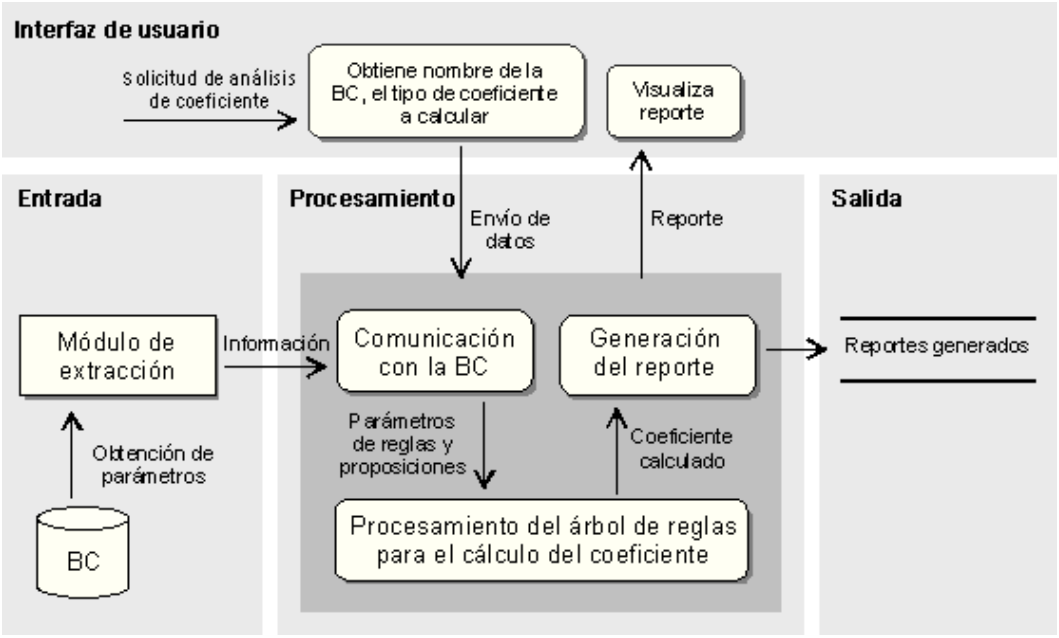

Figura 6.10: Diagrama de contexto del módulo de análisis de coeficientes.

6.4.1. Especificación del módulo de análisis

El método principal del módulo de análisis recibe como datos de entrada: el nombre y ubicación de la BC, la selección del coeficiente(s) a calcular, con la proposición involucrada (para el caso de coeficientes específicos) y brinda como salida: un reporte generado en dependencia del resultado obtenido en la deducción del coeficiente.

Los siguientes son los pasos que se siguen en este módulo en el cálculo del coeficiente para estudiar cualquier BC descrita con HAries:

1. Recibe la ubicación y el nombre de la base.
2. Se especifica el tipo de verificación (por error o por estructura de representación).
3. Se seleccionan los elementos que formarán parte de la verificación (nivel o anomalía específica y estructura(s))
4. Establece comunicación con la base por medio del módulo de extracción.
5. Recibe la meta-información, construida en la etapa de extracción de parámetros, en correspondencia con la elección de los elementos a considerar en el proceso de verificación.
6. Ejecuta los mecanismos de detección de anomalías apropiados.
7. Genera reporte con los problemas localizados.
8. Visualiza el reporte elaborado.

6.4.2. Diseño de la interfaz del módulo de análisis

En la figura 6.11 se presenta la pantalla principal del módulo de análisis de coeficientes.

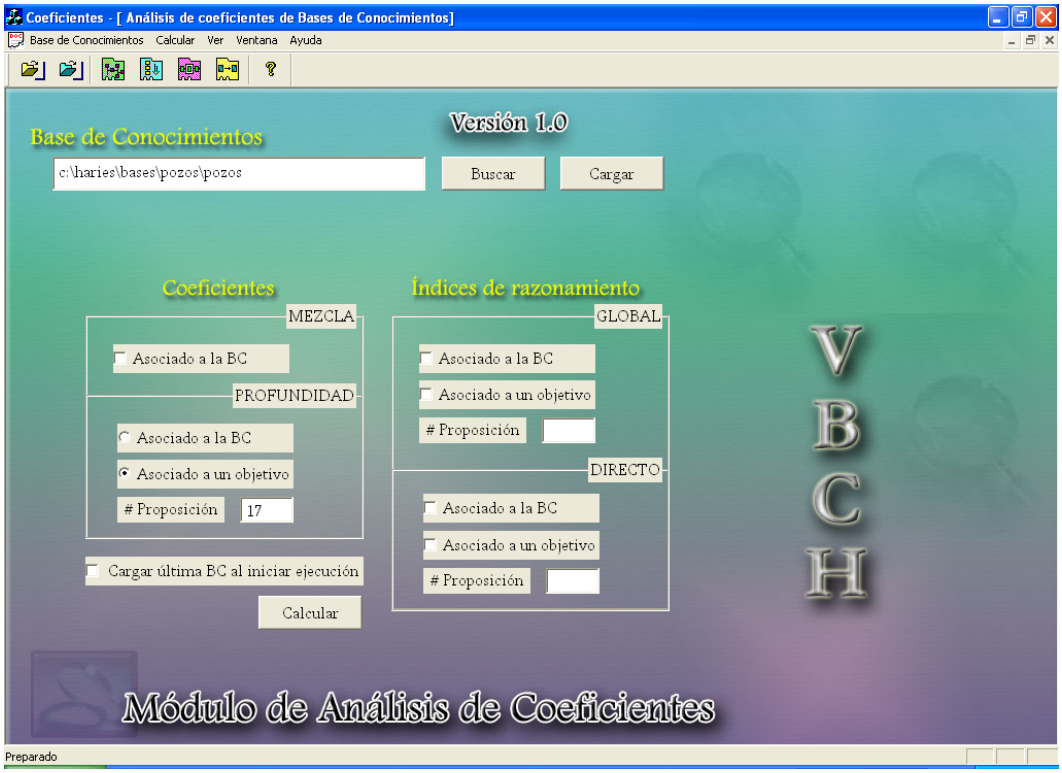

Figura 6.11: Interfaz de usuario del módulo de análisis de coeficientes.

La barra de menús está dividida en cinco grupos de operaciones:

- *Base de Conocimientos*: Contiene las opciones para buscar, abrir la base de conocimientos más reciente o salir de ejecución.
- *Calcular*: Se elige el coeficiente (de mezcla, profundidad, razonamiento global o directo). En todos los casos el análisis que se realiza involucra todas las proposiciones (asociado a la BC).
- *Ver*: Controla la visualización de la barra de herramientas y de estado.
- *Ventana*: Sus funciones permiten la manipulación de todos los reportes elaborados durante una ejecución del sistema.
- *Ayuda*: Proporciona acceso a un hipertexto de ayuda e información de la versión actual del sistema.

La barra de herramientas duplica la órden del menú, *Calcular*.

El primer paso en la ejecución de este módulo consiste en la introducción de la ruta y el nombre de la BC a ser analizada. Seguidamente, se selecciona el coeficiente o índice a ser calculado. El análisis puede incluir todas las proposiciones empleadas en la base de reglas o un objetivo específico.

Una vez que los parámetros para el análisis han sido seleccionados, se presiona el botón *Calcular* que ejecuta el método correspondiente y despliega el reporte generado con las especificaciones del coeficiente obtenido.

6.4.3. Ejemplo de aplicación del módulo de análisis

La figura 6.12 es el resultado de ejecutar el módulo de análisis para la base POZOS y el coeficiente de profundidad para la proposición objetivo 17.

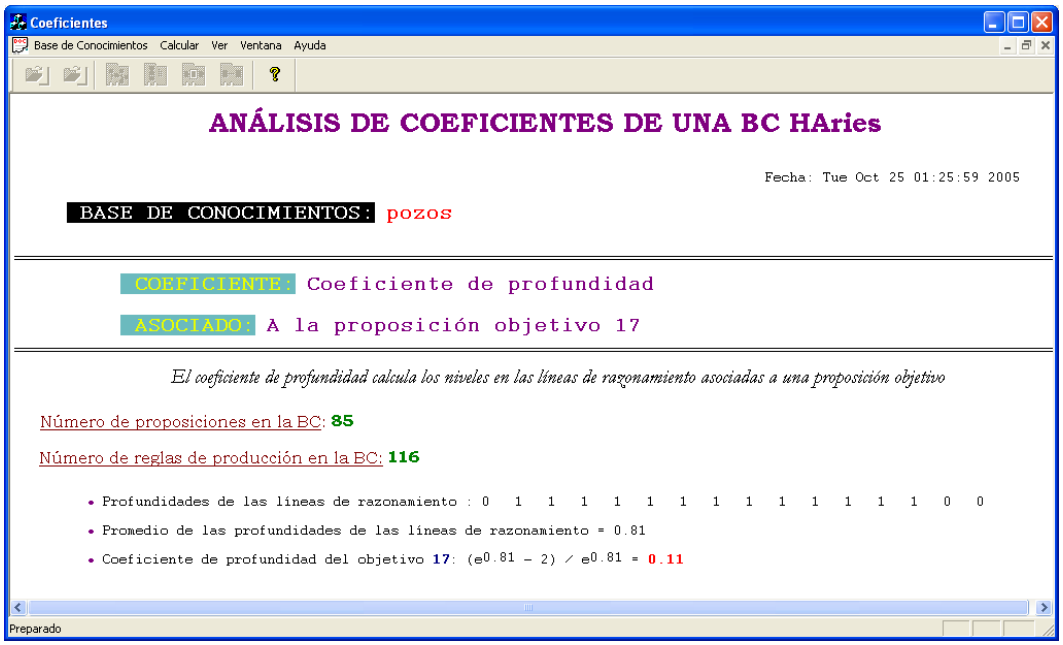

Figura 6.12: Ejemplo de un reporte generado en el módulo de análisis de coeficientes.

En el encabezado del reporte generado se incluye, además del nombre de la base y fecha del análisis, el coeficiente que se ha considerado y el objetivo particular que se analiza. En la parte de especificación se indican las profundidades de las líneas de razonamiento, su promedio y finalmente el valor del coeficiente de profundidad.

Para confirmar el resultado de localizaron el sistema aquellas reglas en la que la proposición $P17$ forma parte del sucedente:

$$R47: \ -32 \ \& \ 16 \ \& \ (-35 \ V \ 36) \Rightarrow 17(0.32 \ 0);$$
$$R83: \ -43 \ \& \ -10 \Rightarrow 17(0.08 \ 0);$$
$$R84: \ 10 \ \& \ 32 \Rightarrow 17(0.34 \ 0);$$

En todas éstas las proposiciones antecedentes son preguntas, excepto $P16$. Es por ello que se reportan tres longitudes 0, correspondientes a las reglas extraídas.

La proposición $P16$ aparece a su vez en el sucedente de trece reglas, en las cuales la totalidad de las proposiciones antecedentes son preguntas, de ahí que el resto de las profundidades tengan longitud 1.

6.4.4. Estructura de clases del módulo de análisis

Para la implementación computacional del módulo de análisis de coeficientes se utilizó la jerarquía de clases de la figura 6.13.

A continuación se describen las clases utilizadas para el cálculo de los coeficientes. Al igual que en el caso anterior, las clases que se encargan del manejo del ambiente son equivalentes a las definidas en el módulo de extracción.

CoefMezcla

Se encarga del cálculo y análisis del coeficiente de mezcla para todos los objetivos de la BC y de la generación del reporte correspondiente.

El constructor recibe el documento de texto donde se escribe el reporte. La función *ReportaCoeficiente* realiza el registro de los resultados en el documento *Reporte* tras la llamada al método *CalculaCoef* (Este proceso es el mismo para todas las clases). Éste ejecuta el conteo de los objetivos con *NumObjetivos* y con *Inferencia* se realiza la extracción de la base informativa para los mismos.

La extracción de la base *Qp* para cada uno de los objetivos se efectúa con la función *BaseInformativa* que implementa el algoritmo 8. Luego para la comparación entre pares de las bases obtenidas se ejecuta el método *CardInterseccion* (recibe las dos bases *Qpi* y *Qpj*).

Los métodos *Ordenar* y *Búsqueda* realizan las funciones que sus nombres indican y son auxiliares en la comparación de los conjuntos correspondientes.

CoefProfundidad

Obtiene el coeficiente de profundidad para uno o todos los objetivos de la BC y reporta los resultados obtenidos.

La función *ProfundidadesLR* implementa el algoritmo 9 para extraer la lista *PLp* de las longitudes de las líneas de razonamiento asociadas a un objetivo específico, mientras que *Inferencia* obtiene los mismos resultados para todos en *PL*. Esta última es llamada

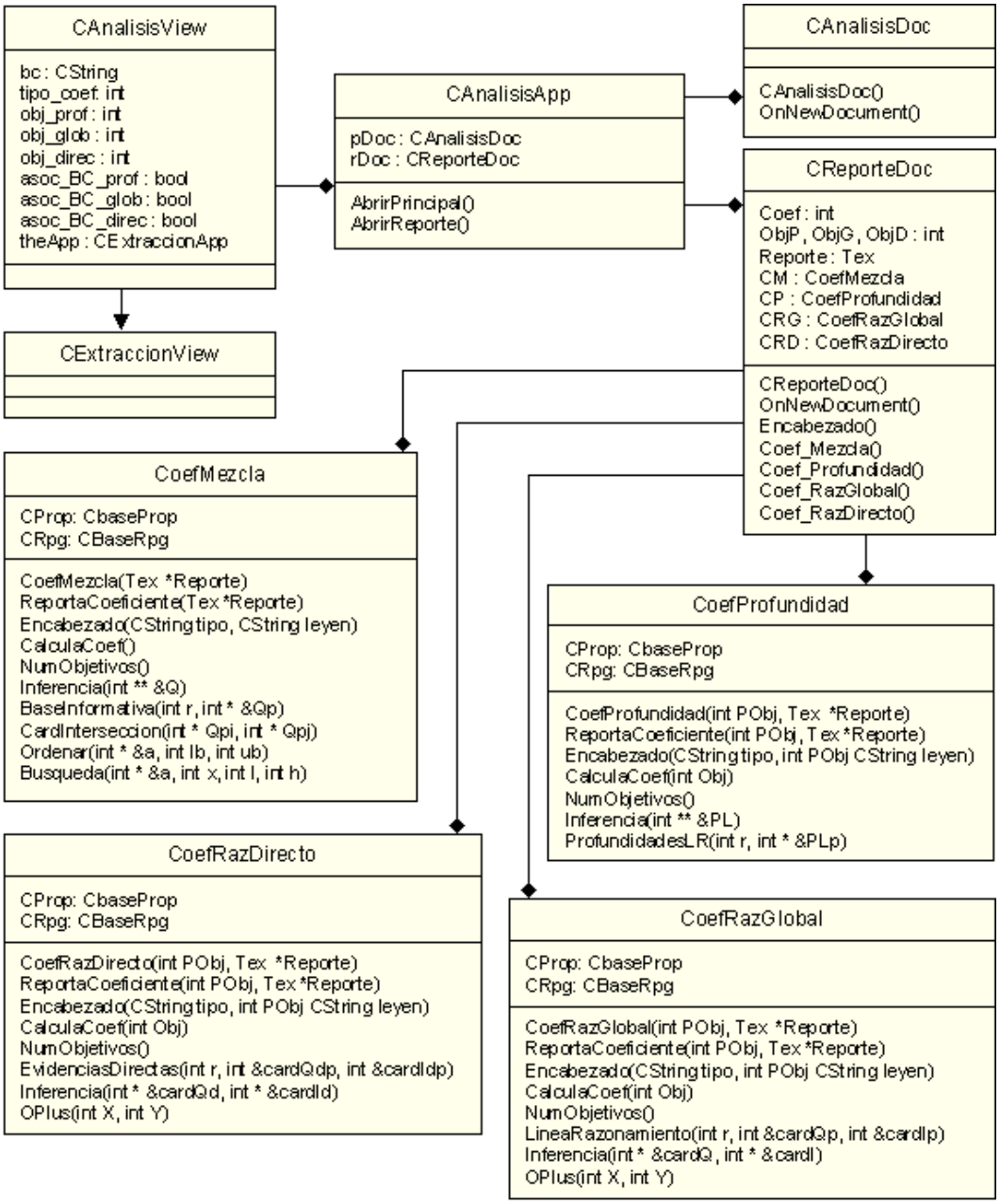

Figura 6.13: Clases para el cálculo de coeficientes.

desde *CalculaCoef* método en el cual se realiza el conteo de los objetivos *NumObjetivos* y se obtiene el valor del coeficiente para el objetivo *Obj*.

CoefRazGlobal

Realiza el cómputo del índice de razonamiento global para una proposición en particular o toda la BC y reporta los resultados alcanzados.

Para este coeficiente, la función *LineaRazonamiento* implementa el algoritmo 10, que además de obtener la cardinalidad base informativa *cardQp*, determina la cantidad de proposiciones intermedios *cardIp* accesibles desde la proposición objetivo que se esté analizando. Al igual que en el resto de las clases, *Inferencia* realiza la extracción para todo el conjunto de objetivos.

El método *CalculaCoef* obtiene el valor del coeficiente para un objetivo en particular *Obj* implementando la definición 28 o para toda la BC. Para el total de objetivos se emplea *Oplus* ($\oplus_c$) como se especifica en la definición 29.

CoefRazDirecto

Calcula el índice de razonamiento directo asociado a una proposición objetivo o la BC y genera el reporte con la solución conseguida.

El método *EvidenciasDirectas*, como su nombre lo indica, extrae las cardinalidades de los conjuntos de proposiciones preguntas *cardQdp* e intermedios *cardIdp* que son evidencias directas de un objetivo específico como se muestra en el algortimo 11 e *Inferencia* de todos los objetivos de la BC.

La función *CalculaCoef* realiza el cálculo del coeficiente para las definiciones 30 y 31 en donde se especifica la forma en que toma valores *Oplus*.

La función *Encabezado* en esta y el resto de las clases se emplea para incluir en el reporte información acerca del *tipo* de coeficiente, el objetivo en análisis y una *leyenda* que explica en qué consiste el coeficiente.

6.5. Conclusiones

En este capítulo se analizó el diseño e implemetación del sistema VBCH, comenzando con su organización general. Además se dedicó una sección a cada uno de los módulos principales de la herramienta: de extracción de parámetros, de verificación y de análisis de coeficientes.

Para cada módulo se especificó su función, sus datos de entrada - salida y el algoritmo que siguen para cumplir con su tarea específica. También se describieron las interfaces y la forma en que se presentan los resultados tomando como caso de estudio un sistema para el pronóstico del efecto de inyectar tenso-activos en pozos de petróleo.

Finalmente se presentaron detalles de la implementación, tomando como base las estructuras de clases para la descripción de los métodos desarrollados. La construcción del sistema se llevó a cabo con el lenguaje de programación Visual C++ en la plataforma Windows, siguiendo las características de implementación de HAries.

Capítulo 7

Conclusiones

En esta tesis se ha descrito el sistema VBCH, que permite la verificación y el análisis de cualquier base de conocimiento que haya sido descrita con el lenguaje de representación HAries [23].

La función principal de VBCH consiste en proveer calidad a aquellos sistemas que sean construidos con HAries y brindar al ingeniero del conocimiento las herramientas para probar el buen diseño del sistema, tomando en cuenta aspectos relacionados con las dos estructuras de representación fundamentales definidas por el lenguaje: proposiciones y reglas de producción generalizadas.

La etapa de *verificación* inicia con la formalización de nociones intuitivas en la comprobación de la BC, introduciendo un marco formal y conceptual para la especificación de la misma. Tal modelo mejora la comprensibilidad de la verificación en términos del conjunto de definiciones de las anomalías que pueden presentarse y los algoritmos empleados para su detección.

En particular, las anomalías que se han considerado tienen que ver con tres propiedades principales del proceso de verificación, éstas son: redundancia, circularidad e inconsistencia. Además se ha incluido el análisis de tipos con la localización de valores ilegales en las estructuras y la definición de anomalías adicionales causadas por la integración de proposiciones y reglas de producción.

El desarrollo en esta etapa se ha inspirado en métodos basados en meta-conocimiento y grafos dirigidos. Del primero se toma la construcción de un modelo partiendo de la extracción de información de la BC, que incluye entre otros parámetros: la separación de las condiciones y acciones de las reglas y los valores máximos de las estructuras.

Los procedimientos de detección de las anomalías del modelo propuesto son efectuados mediante la construcción y examen del árbol expandido por la inferencia del conocimiento. Cada nodo en el árbol representa una proposición, que puede formar parte de la condición o acción en dependencia de la posición que ocupe en la relación;

mientras que los arcos constituyen las reglas que asocian al conjunto de proposiciones.

La separación del proceso de verificación en los métodos descritos permite que los algoritmos propuestos puedan ser empleados en bases construidas con cualquier lenguaje que utilice representación basada en reglas y con nociones de marcos; siempre y cuando se creen técnicas para extraer la meta-información necesaria y ésta sea proporcionada como entrada en dichos algoritmos.

La etapa de *análisis* permite la evaluación de la configuración del conocimiento que está representado en la base a través de las estructuras y específicamente de las relaciones que se establecen entre las reglas de producción.

El concepto fundamental en esta parte tiene que ver con las líneas de razonamiento asociadas a las proposiciones que están presentes en las reglas (ya sea formando parte del antecedente o del sucedente) y el recorrido de las mismas, con el fin de brindar una interpretación del estado o estructura de la BC y advertir sobre las posibles deficiencias que pudieran afectar la ejecución exitosa de un sistema.

Para conseguir esto se han definido un conjunto de medidas o coeficientes que brindan un primer paso en la búsqueda de procedimientos generales para estudiar la composición de las BC usando la arquitectura que proporciona el lenguaje.

En esencia, las medidas se consideraron con respecto a: la complejidad de la base, los niveles de razonamiento que posee y las proporciones entre las proposiciones que son evaluadas mediante preguntas directas al usuario del sistema y las que son deducidas siguiendo el proceso de inferencia.

Como ocurre en la verificación, los coeficientes para el estudio de la composición de las BC y sus interpretaciones se pueden ajustar de forma intuitiva para ser aplicadas a cualquier base de reglas representadas con cualquier lenguaje.

Una vez reseñadas las dos etapas principales, podemos concluir que los resultados principales conseguidos durante la concepción y el desarrollo de VBCH son:

- Se crearon mecanismos que permiten estudiar las BC, por una parte y dotar a los sistemas de posibilidades para detectar errores automáticamente, por otra.

- Se construyeron los algoritmos o procedimientos necesarios para lograr el procesamiento efectivo de las estructuras de representación fundamentales del lenguaje HAries.

- El análisis permite lidiar con reglas que contengan antecedentes complejos y factores de certidumbre.

- La herramienta se puede invocar en cualquier etapa del desarrollo del SBC, ya sea durante la fase adquisición del conocimiento o mantenimiento de la BC.
- El análisis que brinda no está limitado para ejecuciones específicas, por lo cual permite detectar todos los posibles problemas en la BC para las entradas básicas del sistema.
- La etapa de verificación facilita la localización de las anomalías y minimiza su tiempo de detección, sobre todo para desarrolladores de poca experiencia.
- La etapa de análisis brinda orientación en la elaboración de los procesos de razonamiento, cuya representación es difícil y constituye la base fundamental del éxito en la construcción de SBC.

La implementación actual de VBCH ha sido desarrollada con el lenguaje de programación Visual C++ para la plataforma Windows y constituye la unificación, en una herramienta, de tres módulos principales: (1) de extracción de parámetros, (2) de verificación y (3) de análisis de coeficientes. El primero se encarga de la comunicación con la BC y consiste en la interpretación de las estructuras en ella almacenadas para la creación del meta-modelo que se emplea para la detección de las anomalías y el cálculo de los coeficientes. Los dos módulos restantes llevan a cabo las tareas descritas en las etapas de verificación y análisis.

7.1. Trabajos futuros

Como consecuencia del desarrollo y aplicación del sistema VBCH se han identificado ciertas necesidades que brindarían ventajas sobre los resultados alcanzados y servirán para darle continuidad a los mismos:

1. Incorporar nociones extendidas de inconsistencia y redundancia e implementar los métodos requeridos para su detección. Esto es, que el sistema sea capaz de localizar casos en donde dichos problemas involucren más de dos reglas.
2. Identificar y definir formalmente las anomalías locales para el resto de las estructuras de representación del conocimiento que posee el lenguaje HAries y aunado a esto, diseñar algoritmos para el descubrimiento de los errores identificados.
3. Extender los chequeos globales para detectar las anomalías que surjan producto de las relaciones entre las estructuras que son consideradas actualmente y las que sean incorporadas.
4. Integrar los procedimientos implementados de verificación a los compiladores de las estructuras analizadas.
5. Definir nuevos coeficientes que brinden nuevos resultados en cuanto a la composición de la BC y tomando en cuenta el resto de las estructuras.

6. Crear una herramienta de graficación para la visualización y navegación en las dependencias entre proposiciones, partiendo de las relaciones que establecen por medio de las reglas. Esto permitirá el examen de las líneas de razonamiento asociadas a los objetivos de la BC y la detección de problemas de circularidad.

Finalmente, es importante destacar que la herramienta VBCH ha sido incorporada en la etapa de construcción de BC que están siendo desarrolladas actualmente para sistemas con propósitos educativos. Ésto ha garantizado, en cierto modo, la calidad de los mismos, puesto que el ingeniero del conocimiento puede, durante la construcción de dichos sistemas, contar con una forma automática de evaluar su desempeño y confiabilidad, partiendo del estudio de las estructuras principales que posee HAries para la representación del conocimiento mediante los módulos de verificación y análisis.

Su uso, con resultados alentadores, nos ha llevado a plantearnos la tarea de utilizar a VBCH como herramienta de diagnóstico en las etapas de desarrollo de todos los proyectos que sean concebidos de ahora en lo adelante.

Apéndice A

Función de contribución de una regla

La función de contribución a los sucedentes "CTR" completa el análisis de cómo evaluar una regla. Esta depende de los pesos de la regla y del antecedente en doble sentido: primero porque el cumplimiento o no de este último, define con cuál de los dos pesos asociados a cada sucedente trabajar, y segundo, porque su propio grado de certidumbre se necesita para el cálculo, luego se supone que esta función recibe estos datos y retorna una lista de valores que representan el aporte correspondiente a cada proposición sucedente. De esta forma se tendría lo siguiente:

$$\begin{aligned} &CTR(C(A), WS_{11}, ..., WS_{r1}, WS_{12}, ..., WS_{r2}) \\ &= CTRS_1(C(A), WS_{11}, WS_{12}), ..., CTRS_r(C(A), WS_{r1}, WS_{r2}) \end{aligned}$$

Donde $C(A)$ denota la certidumbre del antecedente mientras, que WS_{i1}, WS_{i2} y $CTRSi$ son los pesos y la contribución específica de la regla para el sucedente i.

La función entonces debe trabajar para cada sucedente i por separado y acorde con la definición de RPG debe cumplir las siguientes propiedades:

1. Si se cumple A con completa seguridad, es decir, $C(A) = 1$ entonces $CTRS_i = WS_{i1}$.

2. Si se incumple A con absoluta seguridad, es decir, $C(A) = -1$ entonces $CTRS_i = WS_{i2}$.

3. Si A se satisface con peso no absoluto, $C(A) > 0$ y $C(A) < 1$, entonces se recalcula la contribución usando $C(A)$ y WS_{i1}.

4. Si A se incumple con peso no extremo, $C(A) < 0$ y $C(A) > -1$, entonces se recalcula la contribución usando $C(A)$ y WS_{i2}.

5. Si A es desconocido, es decir, $C(A) = 0$, entonces $CTRS_i = 0$

6. Si A se cumple ($C(A) > 0$) y $WS_{i1} = 0$ o se incumple ($C(A) < 0$) y $WS_{i2} = 0$ entonces también $CTRS_i$ se desconoce, es decir, $CTRS_i = 0$.

Los casos 1 y 2 son claros porque coinciden con la definición de RPG, al igual que 5 y 6, puesto que si no se sabe el peso del antecedente o se conoce, pero no así el peso correspondiente de la regla, entonces esto no brinda información alguna y no puede contribuir. Los casos 3 y 4 constituyen entonces el foco de atención principal.

Para simplificar denotemos como "w" el peso a utilizar como sigue:

$$w = \begin{cases} WS_{i1} & si\ C(A) > 0 \\ \\ WS_{i2} & si\ C(A) < 0 \end{cases}$$

Y como "a" el peso del antecedente de la siguiente forma:

$$a = \begin{cases} C(A) & si\ A\ se\ cumple \\ \\ C(-A) & si\ A\ se\ incumple \end{cases}$$

Luego las propiedades expresadas en 3 y 4 a cumplir por una función CTR expresada en función de a y w son las siguientes:

- Si $w > 0$ entonces $0 < CTRS_i(a, w) <= w$
- Si $w < 0$ entonces $w <= CTRS_i(a, w) < 0$

Dos posibles funciones que cumplen estas propiedades expresadas son las siguientes:

1. Función de Contribución Proporcional.

$$CTRS_i(a, w) = a * w$$

2. Función de Contribución Pura.

$$CTRS_i = \begin{cases} MIN(a, w) & si\ w > 0 \\ \\ -MIN(a, -w) & si\ w < 0 \end{cases}$$

Apéndice B

Función de combinación de reglas

Si en el proceso evaluativo de una proposición intermedio u objetivo existe solamente una regla que aporte contribución (porque es la única donde aparece en el sucedente), entonces el análisis habrá concluido y el peso asociado a la proposición en cuestión es ese exactamente. Si por el contrario hay dos o más reglas con contribución, entonces se necesita combinar todos estos aportes para obtener un resultado único.

Claro está que esto no se puede calcular de forma arbitraria puesto que existen determinadas condiciones que deben cumplirse.

Si $R_1, ..., R_k$ es el conjunto de todas las reglas que tienen la proposición P en el sucedente, al cual se denomina *secuencia de evidencias* asociadas a P y que W_i denota la contribución de la regla $i \in \{1, ..., k\}$ al peso global de P. Entonces la función $GLOB$ es una operación que aplicada sobre todo el conjunto de reglas proporciona el peso de P [25], es decir:

$$GLOB(W_1, W_2, ..., W_k) = W_1 \oplus W_2 \oplus ... \oplus W_k = C(P)$$

donde $\oplus$ denota el tipo de operación.

Entonces si x, y, z denotan contribuciones provenientes de la secuencia de reglas asociadas a una proposición P cualquiera, las propiedades generales que debe tener la función $GLOB$ son [25]:

1. Como el intervalo de pesos definido es [-1,1] la operación tiene que ser interna, es decir $x \oplus y \in [-1, 1]$
2. Asociatividad $x \oplus (y \oplus z) = (x \oplus y) \oplus z$
3. Conmutatividad $x \oplus y = y \oplus x$
4. Existencia de elemento neutro $x \oplus 0 = x$
5. Existencia del opuesto $x \oplus -x = 0, x \notin \{-1, 1\}$

6. Monotonía de la operación $x <= y$ entonces $x \oplus z <= y \oplus z$
7. $x \oplus 1 = 1, x \oplus -1 = -1, x \notin \{-1, 1\}$
8. $1 \oplus -1$ indefinido

Si analizamos las propiedades planteadas veremos que la suma, que es la más simple de las operaciones, podría ser utilizada como operación a no ser porque no es interna en [-1,1]. Ahora bien, como esto es así, lo que se hace es buscar aalguna función f que permita transformar los valores del intervalo (-1,1) en números reales pertenecientes a $(-\infty, \infty)$, para a continuación sumarlos, y después antitransformar el resultado utilizando la inversa f^-1 para obtener de nuevo un número del intervalo original [26]. Simbólicamente tendríamos lo siguiente:

$$x \oplus y = f^{-1}(f(x) + f(y))$$

Uno de los pares de funciones que han sido utilizados en HAries y que coincide con el empleado en el sistema clásico Prospector es [34]:

$$f(x) = \ln \frac{1+x}{1-x}$$
$$f^{-1}(y) = \frac{e^y - 1}{e^y + 1}$$

Para evitar los cálculos de evaluación de la función y su inversa se puede desarrollar la expresión "*", obteniendo la operación:

$$x \oplus y = \frac{x+y}{1+xy}$$

Apéndice C

Proceso evaluativo en HAries

Como se explica en detalle en el capítulo 3 el proceso evaluativo está guiado por los objetivos de la BC. Para ello se emplea el algoritmo 12 que se encarga de la evaluación de las proposiciones. Este algoritmo inicia con la extracción del tipo de proposición ($TipoPro$). En el caso que ésta sea "Pregunta" entonces se evalúa directamente a través del usuario (Preguntar(P)) de lo contrario se encuentra la primera regla Rpg en la cual dicha proposición es parte del sucedente (Saca_1raRegla_esSucedente(P)).

Se inicia el análisis de tal regla mediante el algoritmo 13, cuya salida CTR representa la contribución de dicha regla a la certidumbre de P. Finalmente se calcula mediante la función $GLOB$ la combinación global entre la certidumbre actual de P ($Cert$) y la contribución CTR de la regla. Este proceso continúa mientras se encuentren más reglas con sucedente P (Próxima_regla_esSucedente(P, Rpg)).

Algoritmo 12 Método de evaluación de una proposición

Entrada: La porposición a evaluar, P.
Salida: La certidumbre global de la proposición, $Cert$.

```
Evaluar_Proposición(P)
TipoPro ← Determina_tipo_proposición(p)
if TipoPro = 'Pregunta' then
  Cert ← Preguntar(P)
else
  Rpg ← Saca_1raRegla_esSucedente(p)
  while Rpg > 0 do
    CTR ← Evaluar_Regla(Rpg)
    GLOB(Cert, CTR)
    Rpg ← Próxima_regla_esSucedente(P, Rpg)
return Cert
```

El algoritmo 13 de evaluación de una regla R comienza con la extracción del antecedente Ant y el recorrido por cada proposición p que lo conforma. Estas proposiciones

son evaluadas mediante el algoritmo 12 como se explicó y sus certidumbres son almacenadas en $CertAnt$. Una vez que se tienen todos los pesos, se calcula la certidumbre del antecedente $PesoAnt$ mediante las funciones de evaluación de proposiciones compuestas[1] (Peso_Antecedente($CertAnt$)) y finalmente se aplica la contribución[2] de tal regla a los pesos totales de las proposiciones sucedentes de dicha regla $PesosRegla$.

Algoritmo 13 Método de evaluación de una regla de producción generalizada

Entrada: La regla a evaluar, R.

Salida: La contribución de dicha regla al grado de certidumbre global de la proposición de la cual partió el análisis, CTR.

Evaluar_Regla(R)
$*Ant \leftarrow$Extrae_antecedente(R)
for cada proposición p en el antecedente Ant **do**
  $CertAnt \leftarrow$ Add(Evaluar_Proposición(p))
$PesoAnt \leftarrow$ Peso_Antecedente($CertAnt$)
$CTR \leftarrow$ Contribución_Regla($PesoAnt, PesosRegla$)
return CTR

[1] Véase epígrafe 3.2.5
[2] Véase apéndice A

Bibliografía

[1] M.A. Alonso, A.V. De la Cruz, and A. Gutiérrez. HAries: A knowledge representation language. In *World Multi-Conference on Systemics, Cybernetics and Informatics*. Florida, USA, 2004.

[2] J. Giarratano and G. Riley. *Sistemas expertos. Principios y Programación*. International Thomson, 2001.

[3] S. Shapiro. *Encyclopedia of Artificial Intelligence*. John Wiley and Sons, 1992.

[4] P. Jackson. *Introduction to Expert Systems*. Addison Wesley, 1999.

[5] E. Rich and K. Knight. *Inteligencia Artificial*. McGraw-Hill, 1994.

[6] J.F. Sowa. *Knowledge Representation. Logical, Philosophical, and Computational Foundations*. Brooks / Cole, 1999.

[7] Ch.Y. Suen, P.D. Grogono, and R. Shinghal. Verifying, validating, and measuring the performance of expert systems. *ACM Transactions on Graphics*, 3(11):228–258, 1990.

[8] G.A. Valiente. *Knowledge Base Verification using Algebraic Graph Transformation*. PhD thesis, Universitat De Les Illes Balears, España, 1994.

[9] A. De Antonio, J. Ramírez, and J. Clemente. Crib: A method for integrity constraint checking on knowledge bases. *Revista Iberoamericana de Computación y Sistemas*, 8(1), 2004.

[10] J. Ramírez and A. De Antonio. Checking integrity constraints in reasoning systems based on propositions and relationships. In *13th International Conference on Software Engineering and Knowledge Engineering*. Buenos Aires, Argentina, 2001.

[11] J. Ramírez and A. De Antonio. Knowledge base semantic verification based on contexts propagation. In *Symposium on Model-based Validation of Intelligence*. Carolina, USA, 2001.

[12] P. Meseguer and A. Verdaguer. Verification of multi-level rule-based expert systems: Theory and practice. *International Journal of Expert Systems*, 6(2):163–192, 1993.

[13] D.L. Nazareth and M.H. Kennedy. Verification of rule-based knowledge using directed graphs. *Knowledge Acquisition*, 3(3):339–360, 1991.

[14] R. Nouira and J.M. Fouet. A meta-knowledge based method for detecting invalid, conflictual and redundant rules. In *Workshop on Validation and Verification of knowledge based systems and subsystems*. Oregon, USA, 1996.

[15] J.H. Stephen, J.P Jeffrey, and Ch. Chen. Fuzzy rule base systems verification using high-level petri nets. *IEEE Transactions on Knowledge and Data Engineering*, 15(2):457–473, 2003.

[16] A. Preece. Validation of knowledge-based systems: The state-of-the-art in North America. *Communication and Cognition - Artificial Intelligence*, 11(4):381–413, 1994.

[17] A. Preece. Evaluating verification and validation methods in knowledge engineering. citeseer.ist.psu.edu/515529.html, 2001.

[18] F.P. Coenen. An advanced binary encoded matrix representation for rulebase verification. *Knowledge-Based Systems*, 8(4):201–210, 1998.

[19] J. Santos, C. Ramos, Z. Vale, and A. Marques. VERITAS: An application for knowledge verification. In *IEEE International Conference on Tools with Artificial Intelligence*. Illinois, USA, 1999.

[20] S. Spreeuwenberg and R. Gerrits. Requirements for successful verification in practice. In *The Florida Artificial Intelligence Research Society*. Florida, USA, 2002.

[21] C. Dadkhah and A. Abdollahzadeh. OAV-VVT Expert: An active system for verification and validation of knowledge base based on Ex-OAV KB. *Electronique d'intelligence artificielle*, (15), 2004.

[22] S. Shiu, J. Liu, and D. Yeung. Detection of anomalies of hybrid rule/frame-based expert systems using coloured petri nets. In *Validation, Verification and Refinement of KBS Workshop*. Budapest, Hungary, 1996.

[23] G. Barceló, A.V. De la Cruz, and J.O. Olmedo. VBCH: Herramienta para la verificación de bases de conocimiento. In *3er Congreso Internacional sobre Innovación y Desarrollo Tecnológico*. Morelos, México, 2005.

[24] A.V. De la Cruz and M.A. Alonso. The HAries environment (v6.00) for the development of intelligence systems. *Hífen*, 26(49):184–186, 2002.

[25] A.V. De la Cruz, J.J. Valdés, E. Jocik, J. Balsa, and A. Rodríguez. *Fundamentos y Práctica de la Construcción de Sistemas Expertos*. Academia, 1993.

[26] A.V. De la Cruz. *Representaciones del Conocimiento para la Construcción de Sistemas Expertos con Inteligencia Artificial*. PhD thesis, Universidad de la Habana, Cuba, 1996.

[27] A.V. De la Cruz. Construcción del concepto de sistemas expertos: Ejemplos de aplicaciones. In *Foro Computación de la teoría a la práctica*. D.F, México, 1999.

[28] A.V. De la Cruz and M.A. Alonso. Desarrollo de sistemas computacionales inteligentes híbridos. In *1er Coloquio de Investigación en Ciencias de la Computación*. D.F, México, 2001.

[29] J. Cuena. *Lógica informática*. Alianza Editorial, 1986.

[30] A. D. Preece and R. Shinghal. Foundation and application of knowledge base verification. *International Journal of Intelligent Systems*, 9(8):683–701, 1994.

[31] A.V. De la Cruz, M.A. Alonso, and O. Ramírez. Sistema experto para diagnóstico de transformadores. *Tecnolab*, 14(82):19–32, 1998.

[32] A.V. De la Cruz and M.A. Alonso. Utilización de técnicas de minería de datos e inteligencia artificial para pronósticos en un yacimiento de petróleo. In *Avances en Inteligencia Artificial MICAI/TAINA*. Yucatán, México, 2002.

[33] A.V. De la Cruz, A. Pérez, L.R. Rivera, L. Paz, D. Pozo, and O.H. Cossio. Epilep: An expert system for the diagnosis of epilepsia. In *International Epilepsy Congress*. Brasil, 1991.

[34] J.J. Valdés, A.V. De la Cruz, E. Jocik, J. Balsa, and A. Rodríguez. *Ingeniería del conocimiento en el medio ambiente*. Academia, 1993.

Los abajo firmantes, integrantes del jurado para el examen de grado que sustanterá la **Srita. Grettel Barceló Alonso**, declaramos que hemos revisado la tesis titulada:

Herramienta para el Análisis y Verificación de Bases de Conocimientos descritas con el lenguaje HAries

Y consideramos que cumple con los requisitos para obtener el Grado de Maestría en Ciencias en la especialidad de Ingeniería Eléctrica opción Computación.

Atentamente,

Nombre del jurado 1

Nombre del jurado 2

Nombre del jurado 3