Universidad Politécnica de Madrid
Facultad de Informática

TESIS DE MÁSTER

MÁSTER DE INVESTIGACIÓN EN
INTELIGENCIA ARTIFICIAL

GENERACIÓN DE SISTEMAS BASADOS EN REGLAS MEDIANTE PROGRAMACIÓN GENÉTICA

AUTOR: José María Font Fernández

TUTORES: Daniel Manrique Gamo

Madrid, Junio de 2008

RESUMEN

El objetivo fundamental de esta tesis de fin de máster es la construcción de un algoritmo de generación automática de sistemas basados en reglas mediante técnicas evolutivas, y su aplicación a la resolución del problema de detección de lesiones de rodilla a partir de curvas isocinéticas.

Se presentan dos técnicas diferentes de generación de sistemas basados en reglas a través de programación genética guiada por gramáticas: la primera genera directamente sistemas basados en reglas y la segunda genera indirectamente sistemas basados en reglas difusas representados a través de redes de neuronas difusas. Se introduce un sistema de codificación de individuos específico de cada técnica, una gramática libre de contexto que permite la generación de individuos sujetos a dicha codificación y un método de evaluación de individuos especializado para el problema de detección de lesiones de rodilla.

Asimismo, se presenta un nuevo método de análisis de series temporales de longitud variable que permite convertir una curva isocinética en un vector de dimensión finita, procesable por los generadores automáticos de sistemas basados en reglas.

La aplicación de las técnicas desarrolladas en esta tesis permite la construcción de sistemas basados en reglas y sistemas basados en reglas difusas, a partir de un conjunto de datos de entrenamiento pertenecientes a un dominio de aplicación cualquiera. Estas técnicas permiten la generación de bases de conocimiento de forma automática reduciendo el coste asociado a los métodos tradicionales de educción de conocimientos, los cuales son altamente dependientes del experto del dominio. Los resultados de investigación presentados en este trabajo suponen un avance dentro del área relacionada con la construcción de sistemas inteligentes robustos: sistemas capaces de adaptarse a diferentes dominios o a los cambios que se puedan producir, facilitando el proceso de mantenimiento y actualización constante de una base de conocimiento.

AGRADECIMIENTOS

Dedico este trabajo a mi novia, Isa, a quien debo su amor y apoyo incondicionales desde las trincheras del hogar durante este periplo, al que ahora escribimos juntos un punto y seguido y del que nos quedan incontables páginas por entintar.

Quiero hacer mención especial a la insustituible ayuda en la distancia (y en la proximidad, vía Talgo) de mis padres, José María y Carmen, destinatarios por siempre de toda mi gratitud.

A Rómulo, Isabel y toda su increíble familia, por hacerme sentir como si fueran la mía propia: gracias de todo corazón.

Desde la Universidad, ha sido fundamental para la realización de este trabajo la ayuda de D. Daniel Manrique Gamo, quien me ha guiado en todo momento desde su inicio hasta su finalización. Asimismo, agradezco el apoyo mostrado por D. Juan Ríos Carrión y D. Alfonso Rodríguez-Patón Aradas.

Agradezco también la colaboración de la Dra. África López Illescas, quien ha aportado conocimiento y datos necesarios para la realización de esta investigación.

Mi más sincero agradecimiento y mejores deseos a D. Jorge Couchet Tarantelli, parte esencial de la investigación que ha dado pie a este trabajo.

Gracias a David, Jesús y Marc, compañeros con los que el trabajo deja de ser gris y el café nunca sabe amargo.

Para finalizar, deseo enviar un recuerdo afectivo a Hyoga, pequeño e incansable caballero de los hielos, que ha permanecido a mi lado durante muchas jornadas de trabajo.

ÍNDICE

1. Introducción

Los sistemas de representación del conocimiento formalizan el conocimiento de expertos sobre el dominio de un problema concreto. La ingeniería del conocimiento es el nombre que recibe el área encargada de adquirir y conceptualizar el conocimiento específico de un dominio, formalizándolo en una base de conocimiento [GOME97]. El fin de estos sistemas es su aplicación en el dominio para el cual han sido construidos, aportando una segunda opinión que ayuda a tomar decisiones a un profesional del dominio, por ejemplo un médico encargado de realizar el diagnóstico de una enfermedad.

La representación del conocimiento ha sido ampliamente estudiada por la Inteligencia Artificial (IA) y la Ingeniería del Conocimiento (IC). Su principal objetivo es la creación de formalizaciones del conocimiento que sean procesables por programas informáticos y que se acerquen al modelo de razonamiento humano. Las reglas lingüísticas constituyen una técnica capaz de ejercer de interfaz entre el modo de razonamiento humano y el de procesamiento de un ordenador [PAJA05]. Estas reglas hacen uso de términos lingüísticos y siguen una estructura condicional compuesta por antecedentes y consecuentes, conformando sentencias condicionales del tipo "si X entonces Y", fácilmente comprensibles por un ser humano a la par que procesables por un algoritmo. Estas reglas también se conocen como reglas de inferencia debido a su capacidad de inferir conclusiones a partir del conocimiento actual del entorno.

Las aplicaciones de los sistemas de representación del conocimiento basados en reglas son variadas y usualmente se ubican dentro de los problemas del mundo real: medicina [GOET95], agricultura [MAHA03], control de maquinaria [ANIB04], pronóstico [RAHM96], enseñanza [CHAN95], etc… La mayoría de estas áreas se caracterizan por hacer uso de los sistemas basados en reglas para ayudar a un humano a realizar tomas de decisiones, bien sean diagnósticos médicos o concesiones de seguros. Las áreas más alejadas de la interacción con humanos son las englobadas dentro del control, donde se otorga al sistema el manejo autónomo de un proceso o una máquina. Al contrario que en estos últimos, los problemas englobados en el primer tipo son variables en el sentido en

el que pueden sufrir alteraciones con respecto al tiempo, por ejemplo: aplicado al diagnóstico médico de una enfermedad, un sistema basado en reglas es capaz de dianosticar la presencia o ausencia de dicha enfermedad en pacientes reales observando el estado de una serie de características de dichos pacientes [GOME97]. Dichas características son definidas por una serie de expertos humanos durante el diseño del sistema, atendiendo al conocimiento existente relativo a la enfermedad en cuestión. Cuando dicho conocimiento se actualiza debido a los resultados de una nueva investigación sobre la enfermedad, el sistema desarrollado queda obsoleto dado que estaba preparado para representar el conocimiento desactualizado y diagnosticar con respecto a él. Las líneas de investigación más actuales de la IA intentan resolver este problema con el diseño de sistemas auto-generativos, también llamados sistemas robustos, capaces de transformarse para resolver diferentes problemas así como de evolucionar y adaptarse a la par que éstos cambian. Entre dichas líneas, cabe destacar la Computación Evolutiva (CE).

La CE es un área de la IA que comprende un conjunto de técnicas basadas en los procesos de evolución y selección natural de las especies propuestos por Charles Darwin, así como en los descubrimientos de Gregor Mendel en genética [HOLL75]. La CE se utiliza de forma exitosa en problemas de búsqueda y optimización. Un sistema basado en CE consiste en hacer evolucionar poblaciones de individuos (que representan soluciones de un problema) a partir de una inicial generada aleatoriamente. Dentro de esta rama de la investigación, se encuentran las Estrategias Evolutivas (EEEE), los Algoritmos Genéticos (AAGG) y la Programación Genética (PG) como máximos exponentes.

Las EEEE fueron creadas en la década de 1960 por Ingo Rechenberg, y constituyen un planteamiento para resolver determinados problemas continuos de una forma eficiente [RECH65]. Esos sistemas están basados en técnicas de optimización siguiendo la dirección del gradiente, y dan buenos resultados realizando búsquedas en espacios de soluciones muy amplios. Su principal aplicación se da en problemas de los cuales la información conocida es escasa, al no disponer de expertos capaces de proveer información durante el proceso de diseño del sistema.

Los AAGG, impulsados por John Holland a principios de la década de 1970, son técnicas de búsqueda de soluciones y de optimización inspiradas en conceptos biológicos de la evolución: herencia, mutación, selección y cruce [HOLL69]. Un algoritmo genético (AG), inspirado en el principio de la supervivencia del más fuerte, trabaja con poblaciones de individuos que compiten por sobrevivir adaptándose a las condiciones del medio. Estas condiciones son las impuestas por el problema que se desea resolver, que puede ser cualquiera, a condición de que se disponga del conocimiento suficiente para determinar el grado de adaptación de cada individuo, es decir, para decidir cuáles son los más fuertes de la población. El espacio de búsqueda de un AG son todas las soluciones posibles o fenotipos de un determinado problema codificadas en genotipos de un cromosoma, conformado por una cadena de genes cuyos valores (alelos) tradicionalmente han sido binarios (0 ó 1), ampliados posteriormente a valores reales. Cada cromosoma es representado dentro de la población por un único individuo, siendo la población inicial de individuos generada aleatoriamente. Esta población es sometida durante un número finito de bucles llamados generaciones, a una selección de progenitores que son cruzados para obtener descendencia, que puede sufrir una mutación, y que posteriormente es insertada en la población mediante una operación de reemplazo de individuos. La probabilidad de selección de individuos para el rol de progenitores es directamente proporcional a la proximidad de dichos individuos al óptimo buscado. Los operadores de cruce y mutación permiten renovar la población, expandiendo la búsqueda de la solución óptima fuera de los límites de la población inicial. Los individuos más lejanos del óptimo tienen mayor probabilidad de ser eliminados por el operador de reemplazo.

Las principales desventajas de los AAGG son su baja eficiencia cuando tratan espacios de búsqueda muy grandes, al poder generar individuos inválidos cuya evaluación y eliminación disminuye el rendimiento, así como la estaticidad de sus codificaciones de individuos, que no permiten la variación del tamaño de los mismos durante el transcurso de la ejecución del algoritmo [MANR06].

La PG es una extensión de los AAGG, propuesta por John R. Koza a comienzos de la década de 1990, en la que los individuos codifican programas de longitud variable [KOZA92]. Cada individuo de la población de un programa genético es un programa

capaz de solucionar un problema determinado. Dicha población es sometida, al igual que en un AG, a los operadores de selección, cruce, mutación y reemplazo, debidamente adaptados para su funcionamiento con los individuos de un programa genético.

La Programación Genética Guiada por Gramáticas (PGGG) surge de la aplicación de una gramática libre de contexto (context-free grammar) dentro de un programa genético [WHIG95]. La utilidad de la gramática es la generar el lenguaje de todos los individuos que son posibles soluciones a un problema específico. Cada individuo de la población es una palabra del lenguaje generado por la gramática, y se codifica mediante un árbol de derivación. La PGGG no puede generar individuos inválidos ya que éstos no están contenidos en el lenguaje de la gramática. La PGGG resuelve el problema del cierre [KOZA92], implicando que el cruce de dos individuos válidos genera descendientes válidos, propiedad inexistente en los operadores de cruce de la PG y de los AAGG. El operador de mutación de un programa genético aplicado a un individuo válido debe generar un individuo válido.

Al igual que los AAGG, la PGGG puede ser aplicada a cualquier problema del cual se tenga el conocimiento suficiente para establecer el óptimo a alcanzar y definir los criterios de evaluación de los individuos. Esta cualidad denota la robustez de la PGGG que, aplicada al desarrollo de un sistema de representación del conocimiento para un dominio específico, puede crear sistemas auto-generativos capaces de adaptarse a los cambios en el dominio para el que fueron creados. La representación de los individuos mediante un árbol de derivación usada en los programas genéticos resulta muy apropiada para la codificación de bases de reglas, que son el fundamento de los sistemas basados en reglas.

A tenor de lo expuesto, la propuesta del presente trabajo comprende los siguientes objetivos:

- Realizar un estudio de los sistemas de representación del conocimiento basados en reglas y de su extensión, los sistemas basados en reglas difusas, aplicados a la resolución de problemas del mundo real.

- Exponer el proceso de generación de sistemas basados en reglas mediante PGGG, así como evaluar su eficiencia con respecto a los métodos tradicionales.
- Proponer dos sistemas, uno basado en reglas y el otro basado en reglas difusas, cuyas bases de conocimiento se generen mediante PGGG y aplicarlos al problema de la detección de lesiones de rodilla a partir de curvas isocinéticas.
- Realizar un estudio comparativo de los resultados obtenidos por los sistemas propuestos, valorando las ventajas e inconvenientes de cada uno de ellos.

Se organiza el presente trabajo en capítulos cuyo orden y contenido son expuestos a continuación:

- **Capítulo 2:** Conceptos generales sobre lenguajes y gramáticas según la Teoría de Lenguajes Formales de Noam Chomsky [Chomsky, 1957]. Este capítulo sirve como base para el correcto entendimiento del funcionamiento de la Programación Genética Guiada por Gramáticas.
- **Capítulo 3:** Generación de sistemas de representación del conocimiento basados en reglas: historia, tipos, proceso de diseño y aplicaciones.
- **Capítulo 4:** Generación de sistemas basados en reglas difusas: historia, tipos, proceso de diseño y aplicaciones. Se hace especial hincapié en la hibridación de estos sistemas con Redes de Neuronas Artificiales: Las Redes de Neuronas Difusas.
- **Capítulo 5:** Estado del arte de la Computación Evolutiva, analizando en detalle los Algoritmos Genéticos, la Programación Genética y la Programación Genética Guiada por Gramáticas. Se presentan los diferentes operadores, métodos y técnicas usados en estas disciplinas.
- **Capítulo 6:** Planteamiento del problema de la generación de sistemas basados en reglas mediante técnicas de Computación Evolutiva.
- **Capítulo 7:** Presentación y descripción de los dos sistemas propuestos: la generación automática de un sistema basado en reglas y de un sistema basado en reglas difusas mediante Programación Genética Guiada por Gramáticas.
- **Capítulo 8:** Resultados obtenidos por los dos sistemas presentados en el capítulo 7 tras su aplicación al problema de la detección de lesiones de rodilla a

partir de curvas isocinéticas, incluyendo un análisis detallado de su funcionamiento y una comparativa que esclarece las ventajas e inconvenientes de cada uno de ellos**.**

- **Capítulo 9:** Conlusiones y líneas futuras de investigación deducidas a raíz de la experiencia y los resultados del presente trabajo.

2. Lenguajes y gramáticas

2.1 Introducción

Toda comunicación involucra la utilización de un lenguaje. Las personas se comunican con el resto en diferentes idiomas (lenguajes naturales) o con las máquinas (lenguajes artificiales) a través de conjuntos de símbolos. Se define lenguaje como un conjunto de palabras, también llamadas sentencias o cadenas, que están formadas por símbolos de un alfabeto. Una gramática da cuenta de la estructura de un lenguaje, es decir, de las sentencias que lo forman, proporcionando las formas válidas en que se pueden combinar los símbolos del alfabeto. Una consideración importante es la distinción entre lenguajes formales, que son los que se tratarán en este capítulo, y lenguajes naturales (inglés, español, etc.). La diferencia estriba en que los lenguajes formales (como los lenguajes de programación) obedecen a reglas preestablecidas y por tanto, se ajustan a ellas, no evolucionan y han sido creados para un fin específico. Sin embargo, los lenguajes naturales (utilizados por el hombre) existen "per se" y las reglas gramaticales que rigen su estructura han sido desarrolladas con posterioridad para explicar esta última.

La Teoría de los Lenguajes Formales tiene su origen en un campo aparentemente bastante alejado de la Informática: la Lingüística. Los lingüistas de la llamada escuela estructuralista americana habían elaborado por los años 50 algunas ideas informales acerca de la gramática universal. Se entiende por gramática universal, una gramática que caracteriza las propiedades generales de cualquier lenguaje humano.

El concepto de gramática procede de los estudios de Chomsky [CHOM57] en su búsqueda de una descripción formalizada de las oraciones de un lenguaje natural. Chomsky clasificó las gramáticas en cuatro grandes grupos (G0, G1, G2, G3) cada uno de los cuales incluye los siguientes ($G3 \subset G2 \subset G1 \subset G0$). Las gramáticas tipo 0 se denominan gramáticas sin restricciones o gramáticas de estructura de frases; las gramáticas de Tipo 1 se denominan sensibles al contexto; las gramáticas de Tipo 2 se conocen como gramáticas independientes del contexto o libres de contexto y, por

último, las gramáticas de Tipo 3 se denominan gramáticas regulares. Cada tipo añade restricciones al tipo inmediatamente superior y la jerarquía va desde la más general a la más restrictiva. Cada una de estas gramáticas es capaz de generar un tipo de lenguaje. Un lenguaje L se llama del tipo i (i = 0, 1, 2, 3) si existe una gramática G del tipo i capaz de generar o describir ese lenguaje.

Estos estudios previos sobre teorías de gramáticas formales y lenguajes crearon las bases de la lingüística matemática, la cual tendría aplicación no sólo al estudio del lenguaje natural sino también a los lenguajes formales. Así, los lingüistas distinguen entre gramática particular (propiedades de lenguajes concretos o artificiales) y gramática universal (propiedades generales que pueden aplicarse a cualquier lenguaje humano).

En el campo de la Informática, poco después de las primeras publicaciones de Chomsky, el concepto de Gramática Formal adquirió gran importancia para la especificación de los lenguajes de programación; concretamente, se definió con sus teorías la sintaxis del lenguaje ALGOL 60 (con ligeras modificaciones sobre su versión primitiva), usándose una gramática libre de contexto. Ello condujo rápidamente el diseño riguroso de algoritmos de traducción y compilación.

Finalmente y enlazando con el campo de la lingüística, la Teoría de Lenguajes Formales es de gran utilidad para el trabajo en otros campos de la Informática, por ejemplo en Informática Teórica, Inteligencia Artificial, Procesamiento de lenguajes naturales (compresión, generación y traducción) y Reconocimiento del Habla.

La Teoría de los Lenguajes y Gramáticas Formales tiene una relación directa con la Teoría de Autómatas, siendo posible establecer entre ambas una correspondencia denominada isomorfismo [ISAS97].

2.2 Lenguajes formales

En esta sección se tratan temas relacionados con los lenguajes formales. Se muestran algunas definiciones y operaciones que se pueden realizar con las palabras de un lenguaje así como con los propios lenguajes.

2.2.1 Conceptos básicos

- **Alfabeto:** Un alfabeto (Σ) es un conjunto no vacío finito de elementos indivisibles denominados símbolos (letras, números, combinaciones de letras y números, …)

- **Palabra:** Una palabra es una secuencia finita de símbolos de un alfabeto. Se llama longitud de una palabra x (|x|) al número de símbolos que la componen. La palabra cuya longitud es 0, es decir, no contiene ningún símbolo, se conoce como palabra vacía (λ), $|\lambda| = 0$.

- **Universo de un alfabeto:** Se denomina universo de un alfabeto Σ al conjunto de todas las palabras que se pueden formar con las letras de dicho alfabeto. Se representa como W(Σ). Evidentemente, W(Σ) es un conjunto infinito.

- **Lenguaje sobre un alfabeto:** Un lenguaje sobre un alfabeto (L(Σ)) es todo subconjunto de W(Σ). Como el universo asociado a un alfabeto es infinito, hay infinitos lenguajes asociados a un alfabeto.

2.2.2 Operaciones

- **Operaciones con palabras:**

  - **Concatenación**: Sean dos palabras x e y tales que x Є W(Σ) e y Є W(Σ). Se llama concatenación de ambas palabras, x · y, a la palabra formada por los símbolos de x seguidos de los símbolos de y. A menudo se representa simplemente como xy. Las propiedades de la concatenación son:

    1. Operación cerrada: La concatenación de dos palabras de W(Σ) es una palabra de W(Σ):

    $$x \in W(\Sigma),\ y \in W(\Sigma) \Rightarrow xy \in W(\Sigma)$$

    2. Propiedad asociativa: $x(yz) = (xy)z$
    3. Existencia de elemento neutro: La palabra vacía (λ) es el elemento neutro de la concatenación de palabras, tanto por la derecha como por la izquierda. En efecto, sea x una palabra cualquiera. Se cumple que:

    $$\lambda\, x = x\, \lambda = x$$

    4. La concatenación de palabras no tiene la propiedad conmutativa. A modo de contraejemplo: Sean las palabras x = ABC, y = AD. Es evidente que xy=ABCAD, no es igual a yx= ADABC.

  - **Potencia**: La potencia i-ésima de una palabra x, x^i, se forma por la concatenación i veces de x consigo misma. Por definición, para toda palabra x, se cumple: $x^0 = \lambda$.

  - **Reflexión**: Si la palabra x está formada por los símbolos $A_1\, A_2 \ldots A_n$, , entonces la palabra inversa de x, x^{-1}, se forma invirtiendo el orden de los símbolos en la palabra; $x^{-1} = A_n \ldots A_2\, A_1$.

- **Operaciones con lenguajes:**

  - **Unión**: Sean dos lenguajes definidos sobre el mismo alfabeto, $L_1 \subset W(\Sigma)$, $L_2 \subset W(\Sigma)$. Llamamos unión de los dos lenguajes, $L_1 \cup L_2$, al lenguaje definido como:

    $$L_1 \cup L_2 = \{x \mid x \in L_1 \text{ ó } x \in L_2\}$$

    Es decir, al conjunto formado por las palabras que pertenezcan indistintamente a uno u otro de los dos lenguajes.

    Las propiedades de la unión de lenguajes son:

    1. Operación cerrada: La unión de dos lenguajes sobre el mismo alfabeto es también un lenguaje sobre dicho alfabeto.

    2. Propiedad asociativa: $(L_1 \cup L_2) \cup L_3 = L_1 \cup (L_2 \cup L_3)$.

    3. Existencia de elemento neutro: Cualquiera que sea el lenguaje L, el lenguaje vacío cumple que:

    $$\Phi \cup L = L \cup \Phi = L.$$

    4. Propiedad conmutativa: Cualquiera que sean L_1, L_2 , se verifica que:

    $$L_1 \cup L_2 = L_2 \cup L_1$$

    5. Propiedad idempotente: Cualquiera que sea L, se verifica que:

    $$L \cup L = L$$

  - **Intersección**: Si L_1 y L_2 son lenguajes, su intersección $L_1 \cap L_2$, contendrá todas las palabras que pertenezcan a los dos lenguajes:

    $$L1 \cap L2 = \{x \mid x \in L1 \text{ y } x \in L2\}$$

  - **Resta**: Si L_1 y L_2 son lenguajes, la resta de L_1 y L_2, $L_1 - L_2$, contendrá todas las palabras que pertenezcan a L_1 y no pertenezcan a L_2:

$$L1 - L2 = \{x \mid x \in L1 \text{ y } x \in L2\}$$

- **Concatenación**: Sean dos lenguajes definidos sobre el mismo alfabeto, $L_1 \subset W(\Sigma)$, $L_2 \subset W(\Sigma)$. Se llama concatenación o producto de estos dos lenguajes, L1· L2, al lenguaje definido como:

$$L1 \cdot L2 = \{xy \mid x \in L1 \text{ e } y \in L2\}$$

Es decir, todas las palabras del lenguaje producto se forman concatenando una palabra del primer lenguaje con otra del segundo. Las propiedades de la concatenación de los lenguajes son:

1. Operación cerrada: La concatenación de dos lenguajes sobre el mismo alfabeto es también un lenguaje sobre dicho alfabeto.

2. Propiedad asociativa: (L1L2)L3 = L1(L2L3).

3. Existencia de elemento neutro: Cualquiera que sea el lenguaje L, el lenguaje de la palabra vacía cumple que:

$$\{\lambda\}L = L\{\lambda\} = L.$$

- **Potencia**: Se llama potencia i-ésima de un lenguaje a la operación que consiste en concatenarlo i veces consigo mismo. Por definición, para toda palabra L, se cumple: $L^0 = \lambda$.

- **Clausura positiva**: La clausura positiva de un lenguaje L, se forma por la unión de todas las potencias del lenguaje, excluyendo la potencia 0;

$$L^+ = \bigcup_{i=1}^{\infty} L^i$$

La clausura positiva de cualquier alfabeto (considerado como el lenguaje formado por todos sus símbolos) corresponde al universo del alfabeto excluyendo la palabra vacía (potencia 0 de un alfabeto):

$$\Sigma^+ = W(\Sigma) - \{\lambda\}$$

- **Iteración, cierre o clausura**: El cierre de un lenguaje L se forma por la unión de la palabra vacía a la clausura positiva del lenguaje:

$$L^* = L^+ \cup \{ \lambda \} = \bigcup_{i=0}^{\infty} L^i$$

- **Reflexión**: La reflexión de un lenguaje L está formada por la aplicación de la reflexión a cada una de las palabras del lenguaje:

$$L^{-1} = \{ x^{-1} \mid x \in L \}$$

2.2.3 Reglas

Sea Σ* un alfabeto. Se llama producción o regla a un par ordenado (x,y), donde $x \in \Sigma^*$, $y \in \Sigma^*$. Se dice que x es la parte izquierda de la producción e y la parte derecha. Es frecuente representar las reglas con la siguiente notación:

$$x ::= y$$

Una derivación directa, v -> w, es una aplicación de una producción (x::=y) a una palabra v para convertirla en otra palabra w donde v = z . x .u, w = z . y . u ($v,w,z,u \in \Sigma^*$). Se cumple que, para cada producción (x::=y), existe una derivación directa de x a y: x -> y, lo que se deduce de lo anterior, sin más que hacer $z = u = \lambda$.

Una derivación, $v \rightarrow_* w$, es una aplicación de una secuencia de producciones a una palabra. Una derivación a la izquierda, si se utiliza en cada derivación directa la producción aplicada a los símbolos más a la izquierda de la palabra. Una derivación a la derecha, si se utiliza en cada derivación directa la producción aplicada a los símbolos más a la derecha de la palabra [ISAS97].

2.3 Gramáticas formales

Las gramáticas formales son descripciones estructurales de las sentencias de los lenguajes, tanto formales (lenguajes de programación), como naturales (humanos). Las definiciones proporcionadas en la sección anterior de los lenguajes son extensionales; es decir, para describir el lenguaje se enumeran todas las palabras que lo forman. Las gramáticas permiten describir de forma intencional a los lenguajes. Los lenguajes se describen mediante el alfabeto sobre el que se construyen sus palabras; un símbolo inicial del que parte la obtención de cualquiera de las palabras del lenguaje, denominado axioma; un conjunto de símbolos especiales denominados no terminales que permiten representar subconjuntos del lenguaje o estados intermedios de la generación de las palabras del lenguaje; y un conjunto de producciones, que permiten realizar las transformaciones desde los símbolos no terminales a las palabras del lenguaje. Por lo anterior, pese a que normalmente se dice que una gramática genera un lenguaje, también se dice que una gramática describe un lenguaje.

2.3.1 Conceptos básicos

Se define una gramática formal G como una cuádrupla $G = (\Sigma_T, \Sigma_N, S, P)$, donde:

- Σ_T es el alfabeto de los denominados símbolos terminales.

- Σ_N es el alfabeto de los denominados símbolos no terminales, cumpliéndose:

$$\Sigma = \Sigma_T \cup \Sigma_N \text{ y } \Sigma_T \cap \Sigma_N = \Phi$$

- S es un símbolo no terminal especial ($S \in \Sigma_N$), denominado el axioma de la gramática.

- P es un conjunto finito de reglas (o producciones) que tienen como única restricción que en la parte izquierda de las producciones debe haber al menos un símbolo no terminal, es decir, $P = \{(u::=v) | u = xAy, u \in \Sigma^+, v,x,y \in \Sigma^*, A \in \Sigma_N\}$

2.3.2 Notación de Backus

La notación de Backus es una notación abreviada para representar una gramática. Si el conjunto de producciones contiene dos reglas de la forma:

$$u ::= v$$
$$u ::= w$$

Se representan abreviadamente con la notación:

$$u ::= v \mid w$$

La notación u ::= v | w de las reglas de producción, junto con las reglas de abreviación indicada, se denomina Forma Normal de Backus (o BNF, de las iniciales de la frase inglesa Backus Normal Form) [ISAS97].

2.3.3 Tipos de gramáticas

Noah Chomsky definió cuatro tipos de gramáticas formales, que se diferencian en los tipos de producciones de la gramática. Esta clasificación describe las gramáticas desde los tipos más generales a los más específicos, dependiendo de la forma de las reglas de la gramática. Esta clasificación permite introducir al mismo tiempo, una clasificación en los lenguajes que las gramáticas generan.

- **Tipo 0 (sin restricciones):**

  En la parte izquierda de las producciones tiene que haber al menos un símbolo no terminal. Respecto a las partes derechas no hay ningún tipo de restricción.

  $$P = \{(u::=v)|\ u = xAy,\ u \in \Sigma^+,\ v,x,y \in \Sigma^*,\ A \in \Sigma_N\}$$

  Normalmente, los lenguajes artificiales de los ordenadores digitales no pertenecen estrictamente a este grupo.

Puede demostrarse que todo lenguaje representado por una gramática de Tipo 0 de Chomsky puede ser descrito también por una gramática perteneciente a un grupo un poco más restringido ("gramáticas de estructura de frases"),a cuyas producciones tienen la forma xAy ::= xvy, donde v,x,y $\in \Sigma^*$, A $\in \Sigma_N$. Puesto que v puede ser igual a λ, se sigue que alguna de las reglas de estas gramáticas pueden tener una parte derecha más corta que sea parte izquierda. Si esto ocurre, se dice que la regla es compresora. Una gramática que contenga al menos una regla compresora se llama gramática compresora. En las gramáticas compresoras, las derivaciones pueden ser decrecientes, pues la longitud de las palabras puede disminuir en cada uno de los pasos de la derivación.

- **Tipo 1 (dependientes del contexto o sensibles al contexto):**

Las partes izquierda y derecha de las producciones deben tener una parte común y sólo se admite como regla compresora la regla S::= λ.

$$P = \{(S::= \lambda) \text{ ó } (xAy::=xvy) \mid v \in \Sigma^+, x,y \in \Sigma^*, A \in \Sigma_N\}$$

Se denominan dependientes del contexto porque se tiene en cuenta lo que viene antes y después del símbolo que se sustituye. Así, en la fórmula anterior, x e y son el contexto de la transformación de A en v. Puesto que v no puede ser igual a λ, se sigue que estas gramáticas no pueden contener reglas compresoras. La peculiaridad de estas gramáticas consiste en que las derivaciones producidas por aplicación de sus reglas cumplen que las palabras siempre tienen longitud mayor o igual que las anteriores en la derivación.

$$\alpha_1 \text{ -> } \alpha_2 \text{ -> } \ldots \text{ -> } \alpha_n, \mid \alpha_i \mid \leq \mid \alpha_{i+1} \mid \leq \mid \alpha_n \mid$$

En consecuencia, en una gramática Tipo 1, la palabra vacía λ pertenece al lenguaje representado por la gramática si y sólo si la regla S::= λ pertenece al conjunto de producciones de la gramática.

- **Tipo 2 (independientes del contexto):**

En estas gramáticas, la parte izquierda de las producciones sólo pueden tener un símbolo no terminal:

$$P = \{(S::= \lambda) \text{ ó } (A::=v) \mid v \in \Sigma^+, A \in \Sigma_N\}$$

Estas gramáticas se denominan libres de contexto, porque a la hora de transformar una palabra en otra, el símbolo no terminal que se transforma no depende de los que estén a la izquierda o derecha. Así, cuando se realicen derivaciones en las que se transforme el símbolo A de la fórmula anterior, no hace falta saber qué hay alrededor de él.

La mayor parte de los lenguajes de programación de ordenadores digitales pueden describirse mediante gramáticas de este tipo o muy próximas al mismo.

Definiciones

Dada una gramática tipo 2 (independiente del contexto) se definen:

o **Reglas innecesarias:** Son las que tienen la forma A::=A ($A \in \Sigma_N$). Estas reglas se deben eliminar de las gramáticas ya que no generan derivaciones útiles.

o **Símbolos inaccesibles:** Aquellos símbolos no terminales ($A \in \Sigma_N$) que no pueden ser alcanzados por derivaciones desde el axioma de la gramática, S. Es decir, no hay una derivación $S \rightarrow_* xAy$.

o **Símbolos superfluos:** Su definición depende del conjunto al que pertenezcan. Así:

- Símbolo no terminal superfluo, A: es aquél del que sólo se pueden derivar palabras en las que existe al menos un símbolo no terminal, o, lo que es lo mismo, A #>*x $x \in \Sigma^*T$.

- Símbolo terminal superfluo, a: es aquél que no puede ser alcanzado por derivación desde el axioma, es decir, no existe ninguna producción $A::=\alpha a\beta \in P$.

o **Gramática limpia:** Se dice que una gramática está limpia si no tiene reglas innecesarias, símbolos inaccesibles ni símbolos superfluos.

o **Reglas no generativas:** Una regla es no generativa cuando $A ::= \lambda$, $A \neq S$.

o **Reglas de redenominación:** Una regla es de redenominación cuando A::=B con $A, B \in \Sigma_N$.

o **Gramáticas bien formadas:** Una gramática está bien formada si está limpia, no tiene reglas no generativas y no tiene reglas de redenominación.

Forma normal de Chomsky (FNC)

Las gramáticas de tipo 2 (independientes del contexto), se pueden transformar en gramáticas equivalentes expresadas en la Forma Normal de Chomsky. En esta forma de representar las gramáticas, las producciones pueden tener las siguientes formas:

$$P=\{ (A::=BC) \text{ ó } (S::= \lambda) \text{ ó } (A::=a) \mid A,B,C \in \Sigma_N, a \in \Sigma_T \}$$

Como se puede observar, los árboles de derivación, salvo en las derivaciones correspondientes a las hojas, son binarios. Lo más común es partir de una gramática limpia.

Forma normal de Greibach (FNG)

Otra forma de representar las gramáticas de tipo 2 es por medio de la Forma Normal de Greibach. Esta representación resulta útil para construir el autómata de pila equivalente a una gramática de tipo 2. En este caso, las producciones son de las siguientes formas:

$$P=\{ (A::=aX) \text{ ó } (S::= \lambda) \mid A \in \Sigma_N, X \in \Sigma^*_N, a \in \Sigma_T \}$$

- **Tipo 3 (regulares o lineales):**

  Estas gramáticas, las más restrictivas, pueden ser de dos tipos:

  - Lineales por la izquierda:

    $$P = \{(S::= \lambda) \text{ ó } (A::=Ba) \text{ ó } (A::=a) \mid a \in \Sigma_T, A,B \in \Sigma_N\}$$

  - Lineales por la derecha:

    $$P = \{(S::= \lambda) \text{ ó } (A::=aB) \text{ ó } (A::=a) \mid a \in \Sigma_T, A,B \in \Sigma_N\}$$

Existe, además, una jerarquía entre los tipos de gramáticas tal que las gramáticas del tipo i son más generales que las del tipo i+1. Formalmente $G3 \leq G2 \leq G1 \leq G0$. Cada una de ellas tiene restricciones más fuertes que las anteriores. Las gramáticas de tipo 0 contienen a todas las demás. Las de tipo 1 contienen a las de tipo 2 y tipo 3. Y por último las de tipo 2 contienen a las de tipo 3. Es decir, una gramática de tipo 3 es de tipo 2, de tipo 1 y de tipo 0. Por lo tanto se define una jerarquía de gramáticas respecto de la relación de inclusión, que se pueden representar gráficamente mediante el diagrama de la figura 2.1.

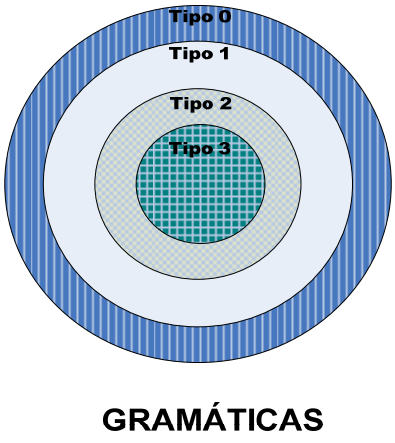


Fig. 2.1: Relación de inclusión entre los diferentes tipos de gramáticas

Se denomina lenguaje de tipo 0 al generado por una gramática de tipo 0. De la misma forma, se denominan lenguajes de tipo 1, tipo 2 y tipo 3, a los generados por las gramáticas de tipo 1, tipo 2 y tipo 3, respectivamente. Si los lenguajes generados por los distintos tipos de gramáticas se relacionan entre sí con respecto a la relación de inclusión se obtiene:

$$\{L(G_3)\} < \{L(G_2)\} < \{L(G_1)\} < \{L(G_0)\}$$

Según lo visto anteriormente existe una correspondencia entre las gramáticas y los lenguajes formales de tal forma que se genera una jerarquía de lenguajes análoga a la mostrada para las gramáticas, que se puede representar gráficamente mediante el diagrama de la figura 2.2 [HOPC02].

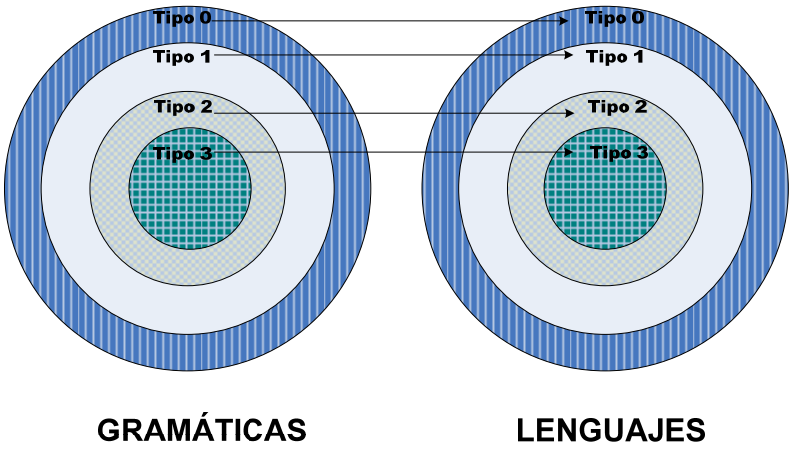


Fig. 2.2: Correspondencia entre gramáticas y lenguajes

2.3.4 Árboles de derivación

Son una forma de representar las derivaciones, siendo utilizados, por ejemplo, en la construcción de compiladores para representar el análisis sintáctico de los programas fuente y sirviendo de base para la generación de código. Sólo se pueden definir árboles de derivación para gramáticas de tipo 1 o más restrictivas (tipos 2 y 3).

Este árbol se construye de la siguiente manera:

a) La raíz del árbol se denota por el axioma de la gramática.

b) Una derivación directa se representa por un conjunto de ramas que salen de un nodo dado. Al aplicar una regla, un símbolo de la parte izquierda queda sustituido por una palabra (x) de la parte derecha. Por cada uno de los símbolos de x se dibuja una rama que parte del nodo dado y termina en otro, denotado por dicho símbolo.

Sean dos símbolos A y B en la palabra x. Si A está a la izquierda de B en x, entonces la rama que termina en A se dibuja a la izquierda de la rama que termina en B.

Para cada rama, el nodo de partida se llama padre, del nodo final. Este último es hijo del primero. Dos nodos hijos del mismo padre se llaman hermanos. Un nodo es ascendiente a otro, si es su padre o es ascendiente de su padre. Un nodo es descendiente de otro, si es su hijo o es descendiente de uno de sus hijos.

A lo largo del proceso de construcción del árbol, los nodos finales de cada paso sucesivo, leídos de izquierda a derecha dan la forma sentencial obtenida por la derivación representada por el árbol.

Se llama rama terminal a aquella que se dirige hacia un nodo denotado por un símbolo terminal de la gramática. Este nodo se denomina hoja o nodo terminal del árbol. El conjunto de las hojas del árbol, leído de izquierda a derecha, da la sentencia generada por la derivación.

Si una gramática es de tipo 3, el árbol correspondiente a cualquier derivación será binario. Es decir, de cada uno de sus nodos no terminales (excepto del último) salen exactamente dos ramas, una de las cuales termina siempre en una hoja. Del último nodo no terminal sale siempre una sola rama.

Por ejemplo, supóngase la siguiente gramática:

$G=(\Sigma_T, \Sigma_N, E, P)$

$\Sigma_T = \{0, 1, [,], +, *\}$

$\Sigma_N = \{E\}$

$P = \{E ::=[EE+], E ::=[EE*], E ::=0, E ::= 1\}$

La figura 2.3 muestra un posible árbol de derivación para obtener la palabra [0[01*]+.

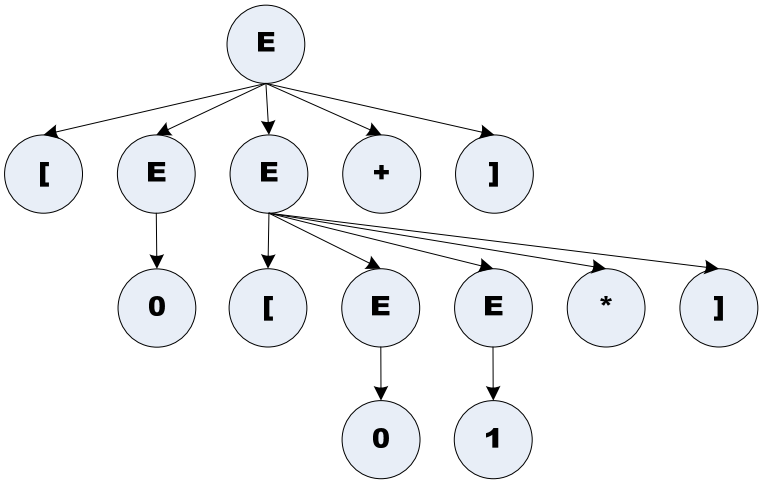


Fig. 2.3: Árbol de derivación para la palabra [0[01*]+ según la gramática G.

Un árbol puede proceder de dos cadenas de derivación distintas:

- Se llama derivación por la izquierda asociada a un árbol, a aquella en la que siempre se deriva primero la primera variable (más a la izquierda) que aparece en la palabra.
- Se llama derivación por la derecha asociada a un árbol, a aquella en la que siempre se deriva primero la última variable (más a la derecha) que aparece en la palabra.

Dado un árbol A correspondiente a una derivación, se llama subárbol de A al árbol cuya raíz es un nodo cualquiera de A, cuyos nodos son todos los descendientes de la raíz del subárbol en A, y cuyas ramas son todas las que unen dichos nodos entre sí en A.

2.3.5 Ambigüedad y recursividad

El concepto de ambigüedad en la teoría de lenguajes y gramáticas es similar al empleado en el lenguaje coloquial: existe más de una forma de generar una palabra a partir del axioma de una gramática. La ambigüedad puede surgir a varios niveles: en sentencias, lenguajes y gramáticas.

- **Sentencia:** Una sentencia es ambigua si tiene más de una derivación o árbol de derivación diferentes.

- **Gramática:** Una gramática es ambigua si tiene al menos una sentencia ambigua. Aunque la gramática sea ambigua, es posible que el lenguaje descrito por ella no lo sea. Puesto que a un mismo lenguaje pueden corresponderle varias gramáticas, aunque una de éstas sea ambigua no quiere decir que tengan que serlo necesariamente las demás. Sin embargo, existen lenguajes para los que es imposible encontrar gramáticas no ambiguas que lo describan.

- **Lenguaje:** Un lenguaje es ambiguo si existe una gramática ambigua que lo genera. Si todas las gramáticas que generan un lenguaje son ambiguas, el lenguaje es inherentemente ambiguo.

El concepto de recursividad en lenguajes y gramáticas también es análogo al utilizado en otras ramas de la computación; la definición de un concepto utiliza a ese mismo concepto en la definición. Existen varios niveles de recursividad:

- **Producción recursiva:** Si el mismo símbolo no terminal aparece en los dos lados de la producción, es decir, existe un $A \in \Sigma_N$ tal que:

$$(A::=xAy) \in P, (x,y \in \Sigma^*)$$

- **Gramática recursiva:** Si existe al menos una producción recursiva en su conjunto de producciones.

- **Recursividad por la izquierda:** Si el símbolo no terminal aparece el primero de la parte derecha, es decir, A::=Ay, ($A \in \Sigma_N$, $y \in \Sigma^*$)

- **Recursividad por la derecha:** Si el símbolo no terminal aparece el último de la parte derecha, es decir, A::=xA, ($A \in \Sigma_N$, $x \in \Sigma^*$)

- **Recursividad por la izquierda en más de un paso:** Si se tiene una regla no recursiva por la izquierda A::=Bx, $A \neq B$, ($A,B \in \Sigma_N$, $x \in \Sigma^*$), pero desde B hay una derivación del tipo $B \rightarrow_* Ay$, ($y \in \Sigma^*$), entonces existe recursividad por la izquierda en más de un paso con respecto al símbolo no terminal A, ya que existe una derivación de la forma: $A \rightarrow_* Ayx$.

2.4 Expresiones regulares

El objetivo de las expresiones regulares es representar todos los posibles lenguajes definidos sobre un alfabeto Σ basándose en una serie de lenguajes primitivos y unos operadores de composición. Lenguajes primitivos son el lenguaje vacío, el lenguaje formado por la palabra vacía y los lenguajes correspondientes a los distintos símbolos del alfabeto. Los operadores de composición son la unión, la concatenación, el cierre y los paréntesis.

2.4.1 Definición de expresión regular

Dado un alfabeto finito Σ, las expresiones regulares sobre Σ se definen como las cadenas sobre el alfabeto $B=\Sigma \cup \{ Ø , \lambda , + , \bullet , * , (,) \}$ que se construyen mediante la aplicación recursiva de las siguientes reglas:

- a $\in \Sigma$, λ y Ø son ER(Σ)

- Si $\alpha , \beta \in ER(\Sigma)$ entonces $\alpha + \beta \in ER(\Sigma)$

- Si $\alpha , \beta \in ER(\Sigma)$ entonces $\alpha \bullet \beta \in ER(\Sigma)$

- Si α , β Є ER(Σ) entonces α * Є ER(Σ) y β* Є ER(Σ)

- Si α , β Є ER(Σ) entonces (α) Є ER(Σ) y (β) Є ER(Σ)

- Solo son expresiones regulares las que resultan de aplicar las reglas anteriores un número finito de veces.

2.4.2 Lenguaje descrito por una expresión regular

Toda expresión regular α sobre Σ representa un lenguaje contenido en Σ*. El operador L define formalmente la relación entre una expresión regular y el lenguaje que representa. L(α) se define recursivamente de la siguiente forma:

- Si α = Ø, L(Ø) = Ø

- Si α = λ, L(λ) = { λ }

- Si α = a, con a ЄΣ , L(a) = {a}

- Si α = α + β, L(α + β) = L(α) + L(β)

- Si α = α • β, L(α • β) = L(α) • L(β)

- Si α = α *, L(α *) = L(α)*

- Si α = (α), L((α)) = L(α)

donde α , β son expresiones regulares.

3. Sistemas de representación del conocimiento basados en reglas

3.1 Introducción

Un Sistema Basado en el Conocimiento (SBC) es aquel en el que aparece representado el conocimiento de un dominio determinado, de tal forma que dicha representación sea procesable por un programa informático. Un Sistema Experto (SE) es un SBC al que se le incorpora conocimiento proveniente de expertos en dicho dominio [PAJA05]. La finalidad de un SE es la resolución de problemas del dominio para el que ha sido creado, aplicando técnicas de razonamiento sobre el conocimiento que alberga su base de conocimiento (BC).

La base de conocimiento es el pilar central de un SBC, ya que es el depósito donde se almacena el conocimiento del que éste dispone sobre un dominio específico. La Ingeniería del Conocimiento (IC) es la disciplina de la Inteligencia Artificial (IA) encargada de realizar hacer explícitos los conocimientos de un dominio en una BC separada del resto del sistema en el que se integra. El ingeniero del conocimiento desempeña la tarea de educir el conocimiento de los denominados expertos del dominio, que son aquellas personas que albergan el conocimiento que se desea representar en una BC, es decir, aquel que es útil para comprender y resolver problemas de un determinado campo de aplicación.

La IA se subdivide en dos vertientes que se complementan: la simbólica, que se basa en el estudio del comportamiento humano ante diferentes problemas para la creación de sistemas inteligentes de propósito específico que emulan dichos comportamientos; y la subsimbólica, cuyo máximo exponente es la IA conexionista, que trata de simular el proceso de cognición humana independientemente de su aplicación a la resolución de problemas concretos. El enfoque del presente trabajo se centra en la vertiente simbólica, cuyas técnicas hacen uso de una representación del conocimiento basada en símbolos. Esta representación está inspirada en la lógica matemática, que hace uso de símbolos

matemáticos para establecer una representación de objetos o ideas del mundo real. Un problema que usa una representación del conocimiento simbólica, toma la forma de una recopilación de símbolos sobre los cuales se aplican algoritmos de procesamiento e inferencia. Una máquina capaz de trabajar con un lenguaje simbólico puede procesar conocimiento representado de manera simbólica.

3.2 Representación simbólica del conocimiento

Un símbolo es una cadena de caracteres que representa una idea o un objeto. Los símbolos son la unidad mínima de representación matemática y lógica del conocimiento, representaciones que están bastante alejadas del tipo de representación de los seres humanos: el lenguaje natural. Una representación simbólica del conocimiento acertada debe ser fácilmente modificable, bien por acción de un ser humano o de un programa automático. Esta cualidad permite la depuración del conocimiento almacenado en una base de conocimiento simbólica, bien mediante la incorporación de nuevo conocimiento o la depuración del ya existente. Se dice que una representación simbólica se hace más abstracta a medida que se aleja del trasfondo matemático para acercarse al lenguaje natural. Dependiendo del grado de abstracción, se distinguen los siguientes tipos de representaciones simbólicas [PAJA05]:

3.2.1 Representación procedural y declarativa

La representación procedural es la forma menos abstracta y más cercana a la representación del conocimiento típica de un ordenador. El conocimiento se almacena en procedimientos enunciados mediante sentencias entendibles por una máquina, de tal forma que la representación del conocimiento y de las instrucciones necesarias para su manipulación se realiza de forma indivisible.

La imposibilidad de separar formalmente el conocimiento y su procesado es ineficiente en el momento en el que el dominio que se desea representar posee incertidumbre relativa a los métodos de procesamiento del conocimiento. Se dice que existe incertidumbre cuando dichos métodos no están matemáticamente respaldados o cuando

la constante evolución del conocimiento del dominio exige una evolución paralela de sus métodos de procesado. La representación declarativa es una extensión de la representación procedural que resuelve este inconveniente, permitiendo la representación por separado del conocimiento y de las técnicas para su procesado. De esta forma, en un dominio con incertidumbre, se pueden ensayar distintas representaciones de conocimiento para resolver uno o varios problemas relacionados de tal forma que, en función al rendimiento y resultados de cada una de ellas, la representación del conocimiento se va refinando hasta alcanzar un alto grado de eficiencia. A la representación simbólica declarativa pertenecen los sistemas basados en reglas o RBSs (del inglés Rule-Based Systems).

3.2.2 Representación relacional

Un escalón por encima de la representación declarativa en términos de abstracción se encuentra la representación relacional, que hace uso de conceptos y relaciones entre conceptos para representar el conocimiento. Este tipo de representación es fuertemente dependiente del tipo de conocimiento que se desea representar.

Los conceptos se representan mediante tuplas de información que contienen atributos específicos para almacenar información. Dependiendo del tipo de información con el que se esté tratando, los atributos y conceptos de la representación relacional varían. Las relaciones entre conceptos almacenan información sobre la interdependencia de conceptos. Este modelo de representación es típico de las bases de datos relacionales.

3.2.3 Representación jerárquica

La representación jerárquica surge como una evolución de la representación relacional, aumentando su grado de abstracción y añadiendo un tipo específico de relación entre conceptos: la herencia. Esta relación permite agrupar conceptos con similitud de atributos, los cuales se heredan de unos a otros creando una estructura jerárquica. Los conceptos más elevados dentro de la jerarquía son los más abstractos, siendo los conceptos inferiores de la jerarquía especificaciones de aquellos conceptos de los que

heredan. Esta amplitud del espectro de abstracción permite a la representación jerárquica procesar información a distintos niveles de profundidad o granularidad. La representación jerárquica es usada por el modelo de representación basada en marcos, así como por los lenguajes de programación orientada a objetos.

3.3 Estructura de un RBS

Un RBS es un sistema basado en el conocimiento en el cual se realiza una representación simbólica declarativa de un dominio mediante reglas de producción o reglas condicionales. Su estructura se muestra en la figura 3.1.

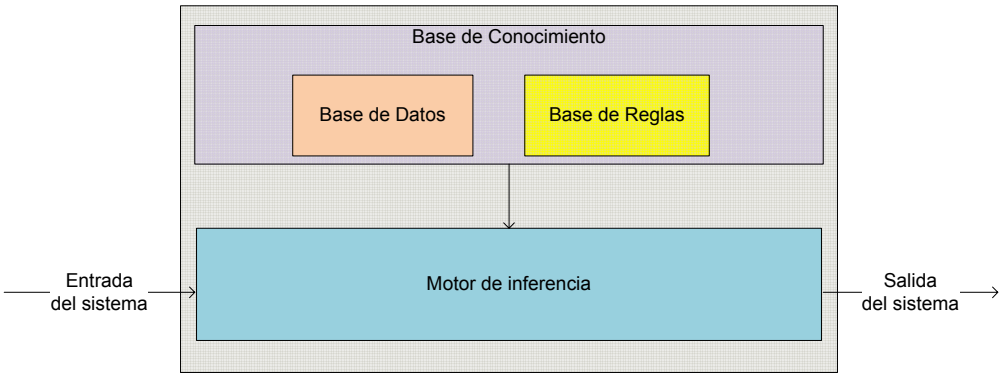


Fig. 3.1. Estructura básica de un RBS.

Los componentes de un RBS son los siguientes:

- **Base de conocimiento:**

  Almacena la representación del conocimiento sobre el dominio de aplicación del sistema. Se divide en la base de hechos y en la base de reglas. En la base de hechos se representa el conocimiento en las variables de entrada y salida del sistema, que forman parte de las reglas semánticas almacenadas en la base de reglas. Una regla semántica es una representación del conocimiento en forma de sentencia condicional. En cada regla se combinan variables de la base de hechos con las partículas condicionales IF y THEN, así como con los operadores lógicos AND y OR. La parte de una regla situada entre IF y THEN es el antecedente de la regla, siendo en resto el consecuente. En el antecedente se

sitúan las premisas que deben cumplirse para que la regla sea aplicable. El consecuente es el conjunto de acciones derivadas de la aplicación de la regla.

- **Motor de Inferencia:**

  Por cada entrada al sistema, el motor de inferencia utiliza el conocimiento almacenado en forma de hechos y reglas para generar una salida. La inferencia en un RBS se basa en técnicas de la lógica proposicional o booleana, donde cada variable (también llamada hecho) es representada mediante un símbolo (X) y únicamente puede tomar dos valores: presente (X) o ausente (¬ X). La expresión más habitual de inferencia en lógica proposicional es el modus ponens:

$$\begin{array}{l} \text{IF X THEN Y} \\ \text{X} \\ \hline \qquad\qquad \text{Y} \end{array}$$

  donde X e Y son variables booleanas, también llamadas bivalentes. En primer lugar, la regla proposicional "IF X THEN Y" indica que, ante la presencia de X se deduce la presencia de Y. A raíz de la posterior observación de la presencia de X, se infiere la presencia de Y. La lógica proposicional permite la inclusión de los operadores lógicos de conjunción ($\wedge$) y disyunción ($\vee$) en los antecedentes de una regla de la forma:

$$\begin{array}{l} \text{IF X} \wedge \text{Y THEN Z} \\ \text{X} \wedge \text{Y} \\ \hline \qquad\qquad\qquad \text{Z} \end{array}$$

  siendo Z una variable booleana. Las variables de un RBS, en lugar de ser bivalentes, toman valores dentro de intervalos acotados definidos en la base de hechos y los operadores AND y OR hacen las veces de los operadores lógicos de conjunción ($\wedge$) y disyunción ($\vee$) respectivamente.

Atendiendo a estos cambios, las reglas de un RBS son como se ejemplifica a continuación:

$$\text{IF } X_1 = 50 \text{ AND } X_2 > 0 \text{ THEN Y = positivo}$$
$$\text{IF } X_1 < 50 \text{ OR } X_1 > 50 \text{ THEN Y = negativo}$$

donde X_1 y X_2 son variables de entrada al sistema e Y es una variable de salida del sistema. De manera análoga al modus ponens, la observación de la variable X_1 tomando un valor igual a 50 y la de la variable X_2 tomando un valor mayor que 0, infieren el hecho de que la variable Y toma el valor "positivo".

3.4 Tipos de inferencia en un RBS

Una regla se activa cuando se satisfacen las cláusulas indicadas por su antecedente, implicando la ejecución de su consecuente. Cuando se activan varias reglas para una misma configuración de las variables de entrada, se genera una salida por cada una de dichas reglas. Dichas salidas son hechos que actualizan el conocimiento actual almacenado en la base de hechos, es decir, que la salida de una regla puede provocar un cambio en el conocimiento que active otras reglas. A este proceso se le conoce como encadenamiento de reglas. Atendiendo al sentido del encadenamiento, se diferencian dos tipos de inferencia: hacia delante y hacia atrás.

3.4.1 Encadenamiento hacia delante

Este tipo de inferencia parte de la observación de hechos en las variables de entrada para, mediante el encadenamiento de reglas, alcanzar un hecho de salida deseado. El proceso es el siguiente:

1. Se define el hecho a alcanzar, es decir, las variables de salida del sistema cuyo valor se desea inferir.
2. Un conjunto de hechos relativo a las variables de entrada es observado por el sistema, es decir, llega una entrada al sistema.

3. Se busca el subconjunto de reglas de la base de conocimiento cuyos antecedentes son satisfechos por los hechos observados.
4. Si el subconjunto está vacío, se finaliza el proceso. En otro caso se continúa en 5.
5. El subconjunto de reglas seleccionado se activa y da lugar a un número de hechos nuevos igual al tamaño del subconjunto.
6. La base de hechos se actualiza con los nuevos hechos.
7. Si se ha alcanzado el hecho de salida deseado se finaliza el proceso, en otro caso, se vuelve a 3.

El encadenamiento hacia delante es típico de sistemas en los cuales se desea conocer el valor de una variable de salida atendiendo a una serie de valores de entrada. Un sistema clásico de encadenamiento hacia delante es PROSPECTOR, que fue aplicado para la predicción de la localización de depósitos de minerales a partir de los datos tomados durante una prospección.

3.4.2 Encadenamiento hacia atrás

El encadenamiento hacia atrás no tiene como objetivo inferir el valor de una variable, sino la demostración de una hipótesis. Su desarrollo se presenta como sigue:

1. Se define una hipótesis, es decir, un valor a alcanzar en una variable de salida del sistema.
2. Si la hipótesis en un hecho comprendido en la base de conocimiento, se finaliza el proceso. En caso contrario se continúa en 3.
3. Se busca el subconjunto de reglas de la base de conocimiento cuyos consecuentes coincidan con la hipótesis.
4. Se establecen los antecedentes de las reglas del subconjunto seleccionado como nuevas hipótesis a demostrar y se vuelve a 2.

Este tipo de inferencia establece (si existe) una cadena de reglas que prueba la veracidad de una hipótesis. MYCIN fue uno de los pioneros en aplicar este tipo de inferencia, con el fin de detectar infecciones bacterianas en pacientes.

3.5 Estrategias de control

Un mismo conjunto de variables de entrada de un sistema puede satisfacer los antecedentes de más de una regla de la base de conocimiento, es decir, varias reglas pueden activarse simultáneamente. El conflicto que surge a la hora de seleccionar qué regla, entre todas las que se han activado, será la primera en escogerse es manejado por la estrategia de control definida para el motor de inferencia. Del orden de selección de las reglas a aplicar depende el tiempo que tarda el sistema de inferencia en alcanzar el hecho de salida, en caso de usar inferencia hacia delante, o la hipótesis a demostrar, en el caso de la inferencia hacia atrás.

No existe una estrategia de control más eficiente que el resto, sino que es recomendable probar con varias hasta encontrar la que mejor se adecúa a las necesidades de cada sistema. Las estrategias más simples son las siguientes:

- Elegir, de entre todas las reglas activadas, aquella que fue introducida con anterioridad en la base de reglas.
- Seleccionar aleatoriamente una de las reglas activadas.
- Contar el número de cláusulas de los antecedentes de cada regla activada y escoger la que dé el resultado más alto, es decir, la más específica.
- Ídem que la anterior, pero contando el número de acciones de los consecuentes de cada regla.
- Seleccionar la regla que utilice el hecho más recientemente deducido.
- Activar un estado finito de letargo para cada regla que haya sido seleccionada, es decir, impedir que vuelva a ser seleccionada hasta dentro de un número predefinido de ciclos de inferencia.

- Evaluar los antecedentes cláusula a cláusula en lugar de hacerlo regla por regla. Hecho esto, establecer un orden de prioridad en la evaluación de las cláusulas de los antecedentes, de tal forma que si una regla no satisface una cláusula, las cláusulas posteriores de dicha regla no han de ser evaluadas, considerando a la regla no activada.

Estas estrategias se asemejan por su sencillez de implementación en el motor de inferencia. Su mayor inconveniente es que están muy lejos de simular las estrategias de resolución de problemas que utilizan los expertos [PAJA05].

Las estrategias de control más complejas añaden prioridad a las reglas de la base de conocimiento. A través del estudio del comportamiento del experto resolviendo problemas del dominio de aplicación, puede educirse el conocimiento que le guía mientras realiza su propio proceso de razonamiento. Siguiendo este enfoque se encuentra la estrategia de control de las meta-reglas, que consiste en crear reglas con un mayor nivel de abstracción (meta-reglas), que guían al motor de inferencia indicándole qué grupo de reglas de la base de conocimiento deben ser evaluadas a cada paso del problema. Estas estrategias se asemejan al modelo de razonamiento de los expertos, a razón de incrementar el coste de implementación del sistema.

3.6 Construcción de RBSs

La construcción de un RBS es llevada a cabo por la figura del ingeniero de conocimiento, comprendiendo dos tareas: la elección del tipo de inferencia a utilizar junto a su estrategia de control y la generación de la base de conocimiento. El tipo de inferencia, con encadenamiento hacia delante o hacia atrás, es seleccionado de manera objetiva en base a cómo las reglas del sistema deben relacionar los hechos con las conclusiones inferidas, lo cual varía según la aplicación del sistema que se desea construir. Cuando las reglas son tales que una conclusión de salida del sistema típica es relacionada con muchos hechos de entrada, se denota amplitud de entrada en el sistema, que es favorable para la aplicación del encadenamiento hacia delante. Sin embargo, si las reglas se comportan de tal forma que a partir de un conjunto típico de hechos de

entrada se alcanzan muchas hipótesis de salida, el sistema demuestra amplitud de salida, apta para el uso del encadenamiento hacia atrás [WINS92]. La elección de la estrategia de control depende de los costes que el desarrollo del sistema puede asumir, seleccionando una estrategia simple en caso de no poderse asumir un coste de implementación alto y viceversa.

Para llevar a cabo la generación de la base de conocimiento se realiza el proceso de adquisición de conocimientos (AC), el cual comprende la extracción de conocimiento de repositorios tales como libros, manuales y bases de datos, así como la educción de conocimiento de expertos [GOME97]. La adquisición es realizada por el IC de manera gradual, descendiendo desde una perspectiva genérica del dominio hasta otra más profunda y específica. En primer lugar, el ingeniero realiza entrevistas abiertas con el experto con el fin de obtener conocimientos genéricos sobre el dominio y familiarizarse con su terminología. En las entrevistas abiertas se interrumpe al experto lo menos posible en sus respuestas, dejándole que se explaye en su discurso siempre y cuando no se aparte de las cuestiones que el ingeniero le plantea. Esta técnica se complementa con la observación directa del modo de trabajo del experto, sin intromisión del IC que se limita a tomar nota.

El siguiente grado de profundidad en la AC supone la extracción de conocimiento mediante el estudio de la documentación disponible. El IC ha aprendido lo suficiente para abordar el estudio y el análisis de textos que versan sobre el dominio, tales como libros, manuales y artículos, generalmente proporcionados por el experto. El objetivo principal es librar al experto de la tarea de formar al ingeniero en el dominio.

Una vez el IC es capaz de obtener información a partir de documentación, el tiempo necesario para la AC disminuye considerablemente. En este punto da comienzo la educción de conocimientos del experto, para la cual se hace uso de las siguientes técnicas: entrevistas estructuradas, incidentes críticos, clasificación de conceptos y emparrillado. En una entrevista estructurada, a diferencia de en su variedad abierta, se espera del experto que responda de manera corta y precisa a las preguntas que el IC le efectúe sobre el dominio. Las cuestiones que requieren distinciones de conceptos muy sutiles o asignaciones de incertidumbre, pueden apartarse de la entrevista y plantearse

en forma de cuestionario que el experto rellena en privado. Cuando la educción de conocimientos mediante la entrevista llega a un punto muerto, la técnica de incidentes críticos sirve como vía de escape al estancamiento, proponiéndosele al experto que describa casos prácticos del dominio. Este recurso proporciona rápidamente información útil para reencauzar la entrevista o para sentar las bases de entrevistas posteriores. La clasificación de conceptos requiere al experto que establezca agrupaciones y jerarquías entre un conjunto de conceptos propuesto por el ingeniero, aportando estructura al dominio. Una vez los elementos del dominio han sido suficientemente estructurados y descritos, se pide al experto que los organice y los plasme mediante una representación personal, que se considera su visión del dominio. La técnica de emparrillado realiza esta representación en forma de matriz bidimensional o parrilla, donde las columnas son elementos del dominio y las filas características de los mismos. Las parrillas esclarecen el conocimiento educido del experto y son fáciles de analizar para encontrar patrones y asociaciones entre los elementos del dominio.

Dentro de la fase de educción del experto, pero a mayor profundidad, se encuentra la técnica del análisis de protocolos, que es una ampliación de la técnica de observación directa en la cual se pide al experto que piense en voz alta mientras resuelve una tarea del dominio. Un protocolo se define como una transcripción de las sesiones de trabajo del experto en la cual se registra qué está haciendo y por qué. El análisis de los protocolos permite identificar conceptos, con sus características y valores, así como las relaciones existentes entre ellos, los distintos estados y objetivos del proceso de trabajo del experto, los operadores que permiten transitar entre estados y las reglas de producción que el experto usa para realizar inferencia.

Para finalizar la fase de educción y el proceso de AC, el ingeniero debe validar los conocimientos adquiridos. Las técnicas más adecuadas para ello son las entrevistas estructuradas y cuestionarios, que proveen encuentros con el experto dirigidos al planteamiento de cuestiones concretas.

Las desventajas comunes en las técnicas de educción son su elevado coste en tiempo, tanto para el IC como para el experto, así como la subjetividad inherente al conocimiento adquirido de este último. Además, las técnicas del emparrillado, la

observación y el análisis de protocolos pueden provocar rechazo por parte del experto, al considerar al ingeniero como un intruso en el dominio. Este hecho puede causar perturbaciones en el proceso normal de trabajo del experto, lo cual introduce ruido en el conocimiento educido. Por su parte, la extracción de conocimiento de documentación escrita está limitada a los textos disponibles en el dominio y es dependiente de la comprensibilidad de los mismos y de la capacidad de entendimiento del ingeniero.

En general, la AC es dependiente de la disponibilidad del experto, ya que requiere tiempo e implicación por su parte, de sus conocimientos, que pueden presentar únicamente una visión parcial del dominio, y de su capacidad de síntesis y de comunicación. Para paliar estos inconvenientes existen técnicas automáticas de extracción de conocimiento a partir de datos, sin mediación de expertos, llamadas técnicas de data-mining o minería de datos [WITT05]. Su finalidad es encontrar agrupaciones de datos con características comunes diferenciadas entre sí dentro de grandes repositorios [LARO06]. Dichas agrupaciones se clasifican en relación a su importancia dentro del dominio del problema y constituyen la base de hechos en forma de variables. Las agrupaciones de estas variables conforman reglas condicionales, las cuales son evaluadas según su aportación al motor de inferencia del RBS para, mediante un proceso de aprendizaje automático, encontrar las reglas óptimas para el dominio del problema.

Siempre que el entorno de desarrollo del RBS lo permite, las técnicas automáticas son combinadas con las técnicas de AC. Esta interacción puede realizarse de dos formas: a través del refinamiento por parte del experto del conocimiento extraído automáticamente mediante minería de datos, o bien evaluando automáticamente el conocimiento educido del experto, a través de algoritmos que realizan la selección óptima de reglas y hechos que conformarán la base de conocimiento del sistema.

3.7 Aplicaciones de los RBSs

Dada su naturaleza, los RBSs son usados como sistemas de propósito específico para realizar tareas de soporte a expertos. Dichas tareas se agrupan de la siguiente forma [GOME97]:

- **Clasificación:**

  Un RBS clasificador proporciona como salida del sistema una respuesta, seleccionada entre un conjunto de respuestas prefijado, conforme a un conjunto de entradas prefijado denominado situación. El RBS empareja cada situación de entrada con una respuesta de salida. Las especializaciones de la clasificación son:

  - **Clasificación jerarquizada:** Se establece una jerarquía de especificidad dentro del conjunto de respuestas a una situación, distinguiendo entre respuestas plausibles, probables e individualizadas.

  - **Interpretación:** El objetivo es analizar información obtenida mediante unos datos de entrada para determinar su significado. En este grupo se clasifican los RBS dedicados a reconocimiento de señales, imágenes, estructuras químicas, etc.

  - **Diagnosis:** Los sistemas encargados de diagnósticos interpretan datos de entrada para detectar irregularidades en los mismos. En medicina, un RBS puede identificar una enfermedad basándose en los síntomas mostrados por un paciente.

  - **Depuración:** Añade a un sistema de diagnóstico la función de indicar, o incluso ejecutar, medidas correctivas ante la anomalía detectada.

- **Predicción y pronóstico:**

  Los sistemas de predicción analizan secuencias de datos distribuidas a lo largo del tiempo para tratar de prever situaciones venideras. Son aplicados a todos los

dominios sujetos al cambio provocado por el paso del tiempo, como la predicción meteorológica, el pronóstico de valores inmobiliarios, etc.

- **Diseño:**

  Un RBS enfocados al diseño de un sistema es capaz de establecer la configuración del mismo atendiendo a sus requisitos, por ejemplo, minimizar una función objetivo que mide los costes de producción de un diseño.

- **Planificación:**

  La planificación consiste en la elaboración de una serie de acciones a seguir para alcanzar un objetivo. En este grupo se engloban los RBSs dedicados a la programación automática, planes de robots, proyectos, cálculo de rutas, etc.

- **Monitorización:**

  Los sistemas de monitorización observan un conjunto de variables de estado correspondientes a un sistema con el fin de alertar ante situaciones anómalas. Un requisito indispensable de los sistemas de monitorización es su funcionamiento constante, ya que su responsabilidad es velar por el buen funcionamiento del sistema monitorizado. Cuando a un RBS de monitorización se le añaden estrategias de reacción ante alertas, se le denomina sistema de control.

- **Ayuda inteligente:**

  Este tipo de RBSs está enfocado a atender las necesidades del usuario de un sistema para proporcionarle consejo e información de interés. Los sistemas de ayuda a la decisión se usan como asesoramiento de expertos en diversos dominios: concesión de créditos, pilotaje de aviones, reparación de sistemas, etc.

- **Instrucción:**

  La finalidad de estos RBSs es instruir a un alumno en una determinada materia, localizando sus errores y ofreciéndole el conocimiento necesario para subsanarlos.

4. Sistemas basados en reglas difusas

4.1 Introducción

Los sistemas basados en reglas difusas (FRBSs, de su nombre en inglés Fuzzy Rule-Based Systems) son una extensión de los sistemas clásicos de representación del conocimiento basados en reglas [CORD01]. Al igual que estos últimos, los FRBSs se componen de reglas condicionales de la forma "IF antecedente THEN consecuente", con la particularidad de que el antecedente y el consecuente son variables de la lógica difusa y no de la lógica clásica. De manera análoga, la lógica difusa se puede considerar una variación de la lógica clásica aplicada a la representación del conocimiento en un entorno de imprecisión, más cercano al modo de representación de conocimiento humano [TRIL92]. Es conveniente aclarar que el término difuso es una traducción del inglés "fuzzy", que en ocasiones también es traducido por borroso o, en menor medida, por vago. Siendo esta última traducción la menos significativa, en este documento se hará uso indistintamente de las dos primeras.

La lógica difusa tuvo sus inicios a raíz de un artículo de Lofti A. Zadeh publicado en 1965, titulado "Fuzzy Sets", que introdujo por primera vez el concepto de conjunto difuso (también llamado partición difusa) como medio de representación del conocimiento [ZADE65]. Esta representación se realiza a través de variables lingüísticas que representan a un conjunto difuso, dentro del cual toman valores dados por una función de pertenencia propia de dicho conjunto. A la hora de realizar una distinción de seres humanos con respecto a la variable lingüística edad, la lógica difusa puede realizar una representación del conocimiento que maneje las etiquetas lingüísticas "Joven", "Adulto" y "Maduro", que conforman tres conjuntos difusos con tres funciones de pertenencia diferentes: toda persona entre 25 y 65 años es adulta, mientras que todo aquel que sea menor de 30 es joven y aquel que sea mayor de 60 es maduro. Estas funciones de pertenencia dejan cierto margen de imprecisión: el existente entre las edades 25 y 30, donde se es joven y adulto al mismo tiempo, y las edades 60 y 65, donde se es adulto y maduro a la vez. La imprecisión o borrosidad es una cualidad

exclusiva de la lógica difusa, también llamada lógica borrosa, en contraposición a la lógica clásica, donde los conjuntos son separados por umbrales que no dan lugar a imprecisión, sino que son nítidos. Esta cualidad hace a los sistemas basados en lógica difusa, como los FRBSs, apropiados para resolver problemas ambientados en dominios donde existe incertidumbre, como son los problemas del mundo real.

Los sistemas de reglas aplicados a problemas del mundo real componen su base de conocimiento mediante dos procesos: la extracción directa de información de repositorios de datos y el aprendizaje de un experto en el dominio de la aplicación. En este último, el experto vuelca el conocimiento que ha adquirido a través de su experiencia para definir, en la medida de lo posible, las variables que conforman el entorno del sistema, las variables internas del sistema y las reglas que definen la lógica del sistema. Los FRBSs facilitan este proceso ya que permiten al experto expresar su conocimiento con variables lingüísticas que son propias del lenguaje humano.

4.2 Conceptos básicos

Se definen a continuación una serie de conceptos que son mencionados a lo largo del presente capítulo, con la intención de aclarar previamente su significado [CORD97] [NGUY97].

- **T-norma:** Una función T: $[0,1] \times [0,1] \rightarrow [0,1]$ es un t-norma sii $\forall$ x,y,z $\in$ [0,1] se verifican las siguientes propiedades:
  - Elemento neutro 1: T (1, x) = x.
  - Monotonicidad: Si $x \leq y$ entonces T (x, z) $\leq$ T (y, z).
  - Conmutatividad: T (x, y) = T (y, x).
  - Asociatividad: T (x, T(y, z)) = T (T (x, y), z).
  - Elemento 0: T (0, x) = 0.

- **T-conorma:** Una función S: $[0,1] \times [0,1] \rightarrow [0,1]$ es un t-conorma $\forall$ x,y,z $\in$ [0,1] se verifican las siguientes propiedades:
  - Elemento neutro 0: S (0, x) = x.
  - Monotonicidad: Si $x \leq y$ entonces S (x, z) $\leq$ S (y, z).
  - Conmutatividad: S (x, y) = S (y, x).
  - Asociatividad: S (x, S(y, z)) = S (S (x, y), z).
  - Elemento 1: S (1, x) = 1.

- **Función de implicación:** Una función continua I: $[0,1] \times [0,1] \rightarrow [0,1]$ es una función de implicación sii $\forall$ x,y,z,x',y' $\in$ [0,1] se verifican las siguientes propiedades:
  - Si $x \leq x'$ entonces I (x, y) $\geq$ I (x', y).
  - Si $y \leq y'$ entonces I (x, y) $\leq$ I (x, y').
  - Principio de falsedad: I (0, x) = 1.
  - Principio de neutralidad: I (1, x) = x
  - Principio de intercambio: I (x, I (y, z)) = I (y, I (x, z)).

- **Operador difuso:** Término designado para realizar operaciones difusas sobre los componentes de un FRBS. Existen tres tipos de operadores difusos: implicación, conjunción y agregación.

- **Conjunto difuso:** Conjunto que puede contener elementos de forma parcial. La pertenencia de un elemento x a un conjunto difuso A viene dada por un grado de verdad $\mu_A(x)$, también llamado grado de pertenencia, de tal forma que:
  - Si $\mu_A(x) = 0$ entonces $x \notin A$.
  - Si $\mu_A(x) = 1$ entonces $x \in$ A totalmente.
  - Si $0 < \mu_A(x) < 1$ entonces $x \in$ A parcialmente.

  Un conjunto difuso se representa mediante una etiqueta lingüística.

- **Variable lingüística:** Cada una de las variables de entrada o salida de un FRBS, que toma valores lingüísticos dentro del conjunto de etiquetas lingüísticas definidas en el mismo.

4.3 Tipos de FRBSs

Los FRBSs se diferencian, atendiendo a su estructura interna, en dos tipos: Mamdani y Takagi-Sugeno-Kang. A continuación se describen ambas variedades, presentando las ventajas e inconvenientes de cada una de ellas.

4.3.1 FRBSs de tipo Mamdani

E.H. Mamdani construyó el primer FRBS que, utilizando la formulación proporcionada por Zadeh, aplicó un sistema de lógica difusa a un problema de control. Esta aplicación de la lógica difusa se conoce como FLC, del inglés Fuzzy Logic Controller o controlador mediante lógica difusa. Su estructura básica, que es muy similar a la de un RBS, se muestra en la Figura 4.1.

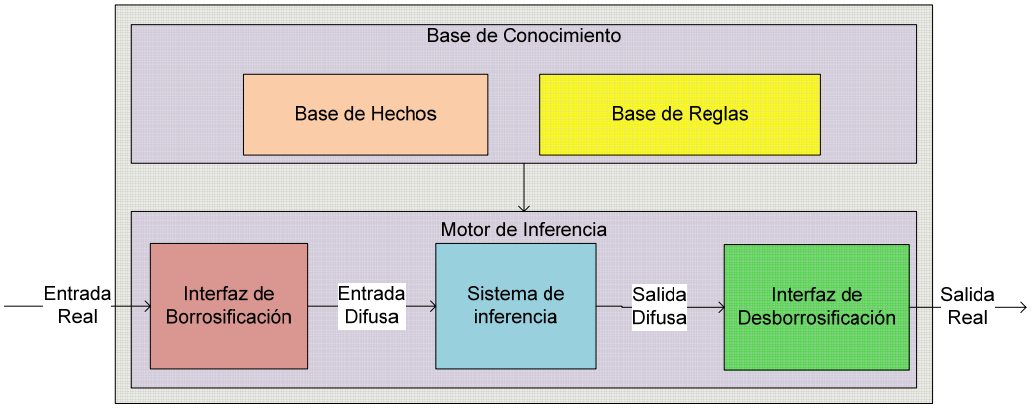


Fig. 4.1. Estructura básica de un FRBS de tipo Mamdani o FLC.

Los componentes del FRBS tipo Mamdani son los siguientes:

- **Base de conocimiento:**
  Almacena todo el conocimiento disponible del problema a tratar en forma de hechos y reglas difusas, a través de los cuales opera el sistema de inferencia para generar una salida difusa para cada entrada difusa. Se divide en una base de hechos y en una base de reglas. La base de hechos contiene un conjunto de

variables lingüísticas de entrada y salida del sistema, así como un conjunto etiquetas lingüísticas, cada una de las cuales define un conjunto difuso. Cada conjunto difuso consta de una función que indica el grado en que una variable lingüística pertenece a dicho conjunto, llamada función de pertenencia. La base de reglas es una colección de reglas condicionales construidas con las variables y etiquetas lingüísticas de la base de hechos, el operador lógico AND y las partículas condicionales IF y THEN. La parte de una regla situada entre IF y THEN es el antecedente de la regla, siendo en resto el consecuente.

- **Motor de inferencia:**

  El motor de inferencia consta de una interfaz de borrosificación, un sistema de inferencia y una interfaz de desborrosificación. La interfaz de borrosificación permite al FRBS traducir entradas reales a sus correspondientes valores en el universo de los conjuntos difusos, con los que opera el sistema de inferencia. El sistema de inferencia genera salidas difusas a partir de las entradas difusas obtenidas de la interfaz de borrosificación, de acuerdo a los hechos y reglas almacenados en la base de conocimiento. Este procedimiento se lleva a cabo a través del modus ponens generalizado, que es una extensión del modus ponens de la lógica clásica, cuyas sentencias condicionales son:

  IF X is A THEN Y is B
  X is A'
  ———————————
  Y is B'

  El modus ponens generalizado adapta estas sentencias a las reglas difusas usadas por un FRBS. De esta forma se tiene que A y A' son conjuntos difusos definidos en U, N es un conjunto difuso definido en V, U es el total de conjuntos difusos de entrada al sistema de inferencia y V es el total de conjuntos difusos de salida del sistema de inferencia. X e Y son variables lingüísticas. Partiendo de esta base, la sentencia condicional "IF X is A THEN Y is B" crea una relación difusa

entre A y B que se define en U x V. Dicha relación R se expresa en un conjunto difuso cuya función de pertenencia es:

$$\mu_R(x,y) = I\,(\mu_A(x), \mu_B(y)), \forall\, x \in U\ , \forall\, y \in V$$

donde $\mu_A(x)$ y $\mu_B(y)$ son los valores resultantes de las funciones de pertenencia de los conjuntos difusos A y B teniendo como entrada X e Y respectivamente. I es un operador difuso de implicación. La función de pertenencia del conjunto difuso B' se obtiene como resultado de la regla composicional de inferencia (Zadeh, 1973): "Si R es una relación difusa definida en U x V y A' es un conjunto difuso definido en U, entonces el conjunto difuso B', inducido por A', se obtiene de la composición de A' y R":

$$B' = A' \text{ o } R$$

Aplicando la regla composicional de inferencia a la i-ésima regla de una base de reglas:

$$R_i : \text{IF } X_{i1} \text{ IS } A_{i1} \text{ AND } \ldots \text{ AND } X_{in} \text{ IS } A_{in} \text{ THEN Y IS } B_i$$

donde n es el número total de variables lingüísticas e i el número total de reglas difusas, la regla composicional de inferencia se reduce a la siguiente expresión:

$$\mu_{B'i}(y) = I\,(\mu_{Ai}(x_0), \mu_{Bi}(y))$$

donde $\mu_{Ai}(x_0) = T(\mu_{Ai1}(x_1), \ldots, \mu_{Ain}(x_n))$, es la entrada difusa del FRBS, $x_0 = (x_1, \ldots, x_n)$ son las variables reales de entrada, T es un operador difuso de conjunción e I un operador difuso de implicación. La salida que genera el sistema de inferencia, que se asigna a la variable lingüística de salida Y, para la regla i-ésima de la base de reglas es el conjunto difuso B'_i junto a su función de pertenencia $\mu_{B'i}(y)$.

Si la base de reglas dispone de m reglas, la salida total del sistema de inferencia es:

$$B'_0, B'_1, \ldots, B'_m$$

La interfaz de desborrosificación agrega la información de estos m conjuntos difusos y la traduce en un valor no difuso correspondiente al dominio de salida del FRBS. El orden de ejecución estos dos pasos, agregación e inferencia, define dos tipos distintos de desborrosificación: FATI (First Aggregate, Then Infer), donde la agregación se realiza previamente a la inferencia, y FITA (First Infer, Then Aggregate), donde se realizan m inferencias cuyos resultados son posteriormente agregados.

Las ventajas de los FRBSs de tipo Mamdani son:

- Al estar formada su base de conocimiento por etiquetas y variables lingüísticas propias del lenguaje humano, facilitan la inclusión de conocimiento de un experto dentro de la base de reglas.
- Poseen un alto grado de flexibilidad en el diseño de las interfaces de borrosificación y desborrosificación, así como en el sistema de inferencia: se puede personalizar un FRBS para su uso en el dominio específico de un problema mediante la selección entre dos métodos de desborrosificación y sus distintos métodos de agregación e inferencia, así como entre las diferentes t-normas que sirven de operadores de conjunción e implicación. Así mismo, las funciones de pertenencia que definen los conjuntos difusos son diseñadas específicamente para cada aplicación.
- Una vez construido el FRBS, la base de reglas es fácilmente interpretable por un humano haciendo comprensible su funcionamiento interno y facilitando la tarea de depuración de reglas por parte de un experto.

Los inconvenientes de estos sistemas son:

- El operador de conjunción AND no aporta potencia suficiente para la elaboración de reglas difusas complejas.
- Los conjuntos difusos establecidos en la base de hechos son rígidos, es decir, no pueden particularizarse para cada regla. Con espacios de entrada de grandes dimensiones, esta división rígida se vuelve ineficiente y hace perder precisión al sistema.
- Si las variables de entrada del sistema son dependientes entre sí, resulta difícil establecer particiones difusas en el espacio de entrada.
- El sistema se hace más preciso a medida que se añaden más variables lingüísticas a la base de hechos. Este aumento hace que el número de reglas crezca, dificultando la interpretación del sistema por parte de un humano.

4.3.1.1 Variantes de los FRBSs de tipo Mamdani

Existen dos variantes de los FRBSs de tipo Mamdani que intentan solventar algunas de las desventajas comentadas:

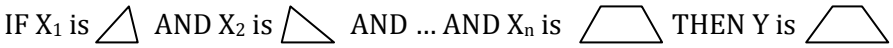
- **DNF FRBSs de tipo Mamdani:**
  Esta variación, cuyo nombre procede del inglés "Disjunctive Normal Form", aporta la inclusión del operador lógico OR a la estructura de las reglas difusas, permitiendo la creación de reglas de la forma:

  IF X_1 IS $\{A_{11}$ OR … OR $A_{1m}\}$ AND … AND X_n IS $\{A_{n1}$ OR … OR $A_{nm}\}$ THEN Y IS B

  donde A_{nm} son etiquetas lingüísticas pertenecientes al conjunto difuso A_n. La primera desventaja de los FRBSs de tipo Mamdani es paliada otorgando mayor flexibilidad y profundidad al proceso de construcción de reglas difusas.

- **FRBSs de tipo Mamdani aproximados:**

  Estos sistemas sacrifican la inteligibilidad de los FRBSs de tipo Mamdani para combatir el segundo y tercer inconvenientes mencionados. La base de conocimiento consta únicamente de base de reglas, eliminando la base de hechos de su interior. Dichas reglas tienen la siguiente forma:



  Las variables lingüísticas ya no son asociadas a una etiqueta lingüística, sino a un conjunto difuso cuya función de pertenencia es definida dentro de la propia regla. Cada regla consta de sus propios conjuntos difusos de entrada y salida, haciéndose cargo de la definición de los mismos y de sus funciones de pertenencia. La precisión de cada regla aumenta al poder definir cada uno de los conjuntos difusos para cada una de las variables lingüísticas que la componen, facilitando la tarea de establecer particiones con variables de entrada interdependientes e impidiendo la pérdida de eficacia del sistema con espacios de grandes dimensiones. Sin embargo, al abandonar el uso de términos lingüísticos la interpretación de estos sistemas es más compleja que la de los FRBSs tradicionales. Además, el aumento de precisión puede conllevar sobreaprendizaje en el entrenamiento del sistema, dando lugar a la pérdida de precisión cuando se aplique a hechos nunca tratados.

4.3.2 FRBSs de tipo Takagi-Sugeno-Kang

Este tipo de FRBSs, también conocido por sus siglas TSK FRBSs, lleva su nombre en honor a los autores del artículo en el que fue presentado originalmente.

La diferencia principal con el modelo propuesto por Mamdani radica en que los consecuentes de las reglas difusas de las que hace uso se representan como una combinación lineal de los antecedentes de dichas reglas.

$$\text{IF } X_1 \text{ IS } A_1 \text{ AND } \ldots \text{ AND } X_n \text{ IS } A_n \text{ THEN } Y = p_1 \cdot X_1 + \ldots + p_n \cdot X_n + p_0$$

donde $\vec{p} = (p_0, p_1, \ldots, p_n)$ es un vector de parámetros reales. El resultado de un TSK FRBS es la media ponderada de las salidas de todas sus reglas difusas, que se calcula de la siguiente forma:

$$y_0 = \frac{\Sigma_{i=1}^{m} h_i \cdot y_i}{\Sigma_{i=1}^{m} h_i}$$

donde m es el número total de reglas, y_i es la salida de la regla i-ésima y $h_i = T(A_{i1}(x_1), \ldots, A_{in}(x_n))$ la correspondencia entre la entrada al TSK FRBS x_0 y el antecedente de la regla i-ésima. T es un operador de conjunción usualmente representado por el mínimo o el producto algebraico.

Los TSK FRBSs ofrecen mayor sencillez de diseño y cálculo que un FRBS de tipo Mamdani, aunque la comprensión lingüística de sus reglas difusas es menos intuitiva debido a la estructura de su consecuente, dificultando la interacción con un experto humano.

4.4 Construcción de FRBSs

La construcción de un FRBS es independiente del tipo elegido, y su proceso consta de dos pasos:

- Diseño del motor de inferencia, que implica la selección entre los operadores difusos disponibles empleados durante el proceso de inferencia.
- Generación de la base de conocimiento, que comprende las variables del dominio de aplicación del sistema de la base de hechos y el conjunto de reglas difusas de la base de reglas.

4.4.1 Diseño del motor de inferencia

Durante el diseño del motor de inferencia para un sistema difuso se deben realizar las siguientes elecciones:

- El operador I usado en el proceso de implicación de las reglas difusas. Existen dos grandes familias que dividen al colectivo de los operadores de implicación: los derivados de la implicación booleana y los derivados de la conjunción booleana. Los operadores más precisos son aquellos pertenecientes a la segunda de estas familias [CORD97], entre los cuales se encuentran las t-normas y las t-conormas. Los más utilizados son la t-norma del mínimo, la del producto y la función de implicación de Lukasiewicz. Existen operadores de implicación fuera de estas familias, pero su utilidad es menor y por ello no se consideran.
- El operador T para la conjunción de los antecedentes de las reglas difusas. La elección más común para este operador es una t-norma, siendo de escasa relevancia cuál de ellas se escoge debido a que muestran similar precisión [CORD97].
- El modo de desborrosificación a usar: FATI o FITA. En el método FATI los operadores de agregación más comunes son la t-norma del mínimo o la t-conorma del máximo y los de inferencia el centro de gravedad o el centro de los máximos. En el método FITA, el resultado de la desborrosificación se obtiene mediante la siguiente expresión:

$$y_0 = \frac{\Sigma_{i=1}^{m} h_i \cdot y_i}{\Sigma_{i=1}^{m} h_i}$$

donde y_i es el valor resultante de la inferencia sobre la regla i-ésima mediante cualquiera de los operadores usados en FATI, y $h_i = \mu_A(x_0)$ la correspondencia entre la entrada al FRBS x_0 y el antecedente de la regla i-ésima. Ambos métodos muestran buenos resultados [CORD97], siendo FATI el método original propuesto por Mamdani para el primer FLC. Sin embargo, el método FITA resulta ventajoso cuando el problema al que se aplica el FRBS necesita rapidez

de respuesta, ya que evita realizar la operación de agregación sobre las salidas difusas de las reglas, suponiendo un ahorro computacional frente al método FATI.

4.4.2 Generación de la base de conocimiento

El primer paso a realizar en la generación de la base de conocimiento de un FRBS es la selección de las entradas y salidas reales de la base de hechos. De entre todas las entradas y salidas reales existentes dentro del dominio en el que se va a implantar el sistema, se escogen aquellas que son relevantes para el funcionamiento del mismo. Las variables reales escogidas definen las variables lingüísticas de entrada y salida que forman parte de los antecedentes y consecuentes de las reglas difusas del sistema.

Definidas las variables lingüísticas, se debe definir los conjuntos difusos a los cuales estará relacionada cada una de ellas, atribuyendo a cada conjunto una etiqueta lingüística y una función de pertenencia. El alcance y el número de los conjuntos difusos definen la granularidad del sistema. Las funciones de pertenencia se diferencian entre sí con respecto a su forma, siendo usuales las funciones triangulares, trapezoidales, gaussianas y exponenciales. De entre todas ellas, las funciones trapezoidales demuestran ser buenas aproximaciones de las gaussianas y exponenciales, que son más precisas pero computacionalmente más costosas. En el caso de los FRBSs aproximados no se debe establecer ninguna etiqueta lingüística ya que estos sistemas no hacen uso de ellas.

Por último se construyen las reglas difusas que conformarán la base de reglas del sistema. El proceso de adquisición de conocimientos usado en la construcción de la base de conocimiento de los RBSs es también aplicable a los FRBSs. Al ser la representación del conocimiento de un FRBS más próxima al modo de razonamiento del experto, dada la inclusión del componente difuso en la base de conocimiento, la educción de conocimientos se hace más sencilla.
Sin embargo, dependiendo del tipo de FRBS elegido, el proceso de adquisición de conocimientos varía:

- **Generación de reglas lingüísticas:** Las reglas lingüísticas son aquellas usadas por los FRBSs de tipo Mamdani simples o en su variante DNF. Debido a su estructura puramente lingüística, este tipo de reglas es el ideal para ser generado a partir del conocimiento derivado de un experto en el dominio de aplicación del sistema. Si un experto posee conocimiento suficiente sobre el dominio, es capaz de definir la totalidad de la base de conocimiento del sistema: hechos y reglas. El experto expresa su conocimiento en términos lingüísticos que son directamente traducidos en variables, etiquetas y reglas lingüísticas. En las ocasiones en que la información aportada por el experto es incompleta, se hace uso de métodos de minería de datos para definir las variables y etiquetas lingüísticas, mientras que para la obtención de reglas se aplican métodos de aprendizaje automático. La interacción entre el conocimiento del experto, la minería de datos y el aprendizaje automático suele ser de refinamiento: se generan variables, etiquetas y reglas automáticamente que un experto refina posteriormente, o bien un experto establece una base de conocimiento que es completada y refinada por métodos automáticos.

- **Generación de reglas aproximadas:** Las reglas aproximadas hacen referencia a las usadas por los FRBSs de tipo Mamdani aproximados. En este tipo de sistemas, la aplicación del conocimiento de un experto es limitada a la definición de las variables lingüísticas debido a la no existencia de etiquetas lingüísticas. Debido a esto, un experto no es capaz de formular lingüísticamente reglas aproximadas. Sin embargo, es posible realizar la traducción de un sistema de reglas lingüísticas a un sistema de reglas aproximadas, sustituyendo cada etiqueta lingüística por una función de pertenencia aproximada, mediante métodos matemáticos, a la función de pertenencia del conjunto difuso al que representa.

- **Generación de reglas TSK:** Estas reglas son las usadas por los TSK FRBSs. La no existencia de consecuentes lingüísticos en estos sistemas limita la aplicación del conocimiento de un experto a la definición de etiquetas lingüísticas de entrada y de variables lingüísticas de entrada y salida. La generación de las

combinaciones lineales de antecedentes que conforman los consecuentes de las reglas TSK se realiza mediante la aplicación de métodos de aprendizaje automático. Este tipo de sistemas no es muy útil cuando se dispone de un experto capaz de aportar bastante información sobre el proceso de diseño del FRBS.

4.5 Aplicaciones de los FRBSs

Los tres campos de aplicación fundamentales de los FRBSs son el modelado, el control y la clasificación [CORD01].

4.5.1 Modelado difuso

Se entiende por modelado difuso a la tarea de aproximar una función continua mediante un FRBS. En contraposición a los algoritmos clásicos de aproximación de funciones, como la regresión o las redes neuronales, en los que la comprensión del resultado de una aproximación no es intuitiva, los FRBSs ofrecen al usuario una clara interpretación de la aproximación de una función. Para esta tarea, cada regla de un FRBS representa un aproximación local de la función total fácilmente interpretable y analizable. Estas aproximaciones locales aumentan la precisión del modelado, aunque elevan el número de parámetros a ser estimados de manera global. La aproximación local confiere mayor robustez al modelado, ya que la variación de una regla difusa únicamente afecta a la región local aproximada por dicha regla, no al modelo completo de la función.

Las aplicaciones más comunes del modelado difuso a problemas del mundo real son:

- **Ciencias económicas:** Predicción de tasas de cambio de moneda para actuar de sistema de ayuda a la decisión en el mercado de divisas, aproximación de series temporales financieras y detección de anomalías en las reclamaciones de seguros sanitarios.

- **Predicción del tiempo:** Modelado de la temperatura media diaria.

- **Medicina:** Extracción de conocimiento de diagnósticos médicos para la elaboración de clasificadores que actúen de sistemas de ayuda a la decisión en casos clínicos.

- **Ingeniería:** Resolución de problemas de distribución eléctrica, estimación del número de líneas de bajo voltaje existentes dentro de una instalación y del coste de mantenimiento de dichas líneas.

4.5.2 Control difuso

El control difuso supone el manejo de maquinaria industrial por parte de un FRBS, que es capaz de aprovechar toda la información disponible proveniente de sensores, mediciones y conocimiento experto sobre una máquina concreta, para proveer descripciones lingüísticas e instrucciones de control comprensibles sobre la misma. El control difuso es independiente de la máquina a la que se aplica, por lo que se facilita el diseño del controlador al no necesitar un modelo matemático del funcionamiento de dicha máquina. Los controladores difusos son aplicados en tiempo real, y su funcionamiento básico es la continua observación de un conjunto de variables de estado de la máquina controlada y la emisión de una salida de control al respecto.

La primera aplicación industrial de un controlador difuso fue para un horno de cemento en Dinamarca en 1979. Su uso se ha extendido desde entonces a diversos campos de la automatización industrial: plantas de tratamiento de agua, incineradores, fermentación de sake, ascensores, ventilación de túneles, lavado de coches, lanzaderas espaciales, reactores nucleares, conducción de trenes y robótica.

4.5.3 Clasificación difusa

El objetivo de la clasificación de patrones es el de asignar una clase a un objeto descrito por un vector de características. La construcción de un sistema clasificador tiene dos pasos: el entrenamiento y el testeo del sistema. Cuando el sistema está en fase de entrenamiento, aprende a clasificar un conjunto de patrones cuya correcta clasificación es conocida previamente, llamados patrones de entrenamiento. Finalizado este paso el sistema entra en la fase de testeo, en la cual ha de usar el conocimiento aprendido en el entrenamiento para clasificar patrones que no ha tratado anteriormente. Las técnicas clásicas de construcción de clasificadores son las redes bayesianas, los árboles de decisión y las redes neuronales. La ventaja de los clasificadores difusos sobre estas técnicas es que imitan el proceso de clasificación en entornos imprecisos que sigue un ser humano, capaz de reconocer objetos a pesar de encontrarse en un entorno con falta de información.

Los clasificadores se dividen en dos grupos dependiendo de su uso: los que trabajan de manera autónoma y los que sirven como sistema de segunda opinión a un experto humano. Los clasificadores del primer tipo únicamente deben medirse por su precisión a la hora de realizar clasificaciones. Sin embargo, en el segundo tipo prima la comprensión por parte de un humano del modo de razonamiento que sigue el sistema, la robustez y la versatilidad del mismo, así como la coherencia con el conocimiento previo del experto sobre el entorno de aplicación. Un FRBS es capaz de, a través de sus reglas, proporcionar un proceso de clasificación transparente al experto, fácilmente comprensible e incluso refinable mediante su conocimiento adquirido a través de la experiencia.

Los clasificadores difusos pueden generar tres tipos diferenciados de reglas:

- **Reglas difusas con una clase en el consecuente:** Estas reglas tienen la siguiente estructura:

$$\text{IF } X_1 \text{ IS } A_1 \text{ AND } \ldots \text{ AND } X_n \text{ IS } A_n \text{ THEN Y IS } C_i$$

donde X_1, …, X_n son las características del vector que define a cada patrón, Y es la variable lingüística de salida, A_i, …, A_n son las etiquetas lingüísticas que representan a los conjuntos difusos almacenados en la base de hechos del sistema y C_i (i = 1, …, k) es la etiqueta de la clase asignada a cada patrón entre un total de k clases.

- **Reglas difusas con una clase y un grado de certeza en el consecuente:** Estas reglas son similares a las anteriores, con el añadido de un grado de certeza a la clase adjudicada a cada patrón:

$$\text{IF } X_1 \text{ IS } A_1 \text{ AND } \ldots \text{ AND } X_n \text{ IS } A_n \text{ THEN Y IS } C_i, r_i$$

  El grado de certeza indica la pertenencia del patrón clasificado a la clase C_i.

- **Reglas difusas con un grado de certeza en el consecuente para cada clase:** En estas reglas no se asigna ninguna clase en concreto a un patrón, sino que se presenta como consecuente un vector de grados de certeza:

$$\text{IF } X_1 \text{ IS } A_1 \text{ AND } \ldots \text{ AND } X_n \text{ IS } A_n \text{ THEN } (r_1, \ldots, r_k)$$

  donde r_i es el grado de certeza del patrón a la clase C_i.

Los problemas de clasificación del mundo real más usuales a los que se aplica un FRBS son:

- **Reconocimiento de imágenes:** Segmentación de mapas geográficos en componentes (calles, carreteras, campo, etc…) y clasificación de superficies mediante fotografías.

- **Reconocimiento de caracteres:** Reconocimiento de caracteres alfanuméricos manuscritos.

- **Medicina:** Diagnóstico de infarto de miocardio y de cáncer de mama.

- **Clasificación del tiempo:** Clasificación de la cantidad de precipitación, temperatura y viento diarias para realizar un modelado de la precipitación en una zona.

4.6 Hibridación de FRBSs con redes de neuronas artificiales

Las redes de neuronas artificiales (RNA) carecen de flexibilidad y de capacidad para representar el conocimiento aprendido, mientras que los FRBSs no son capaces de aprender ni de realizar computación en paralelo [TRIL92]. La hibridación entre los FRBSs y las RNAs tiene como fin la creación de sistemas en los que se integran las capacidades de adaptación y de aprendizaje de las redes neuronales con la facilidad de interpretación propia de las reglas difusas. Los sistemas surgidos de esta integración se diferencian en dos clases: sistemas neuro-difusos y redes de neuronas difusas.

4.6.1 Sistemas neuro-difusos

Los sistemas derivados de este tipo de hibridación utilizan la capacidad de interacción con expertos de los FRBSs para construir un sistema de reglas difusas que más tarde es depurado y optimizado mediante una RNA [TRIL92]. Esta optimización se puede llevar a cabo de dos maneras diferentes: incluyendo una RNA para modificar la forma de las funciones de pertenencia del FRBS y hacerlas más precisas, o traduciendo el FRBS completo en una topología neuronal que refleja el conocimiento extraído del experto. En ambos casos, los FRBSs son usados a modo de interfaz para recabar conocimiento que más tarde es depurado por una RNA [ZAHE06].

4.6.2 Redes de neuronas difusas

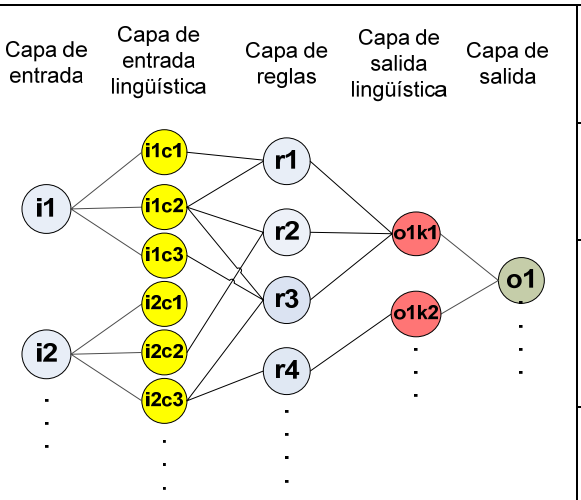
Las redes de neuronas difusas o FNN (del inglés Fuzzy Neural Network), implementan la representación difusa del conocimiento propia de los sistemas difusos dentro de una topología neuronal, introduciendo borrosidad en las entradas y salidas de una red neuronal [CARR03].

Una FNN implementa operadores lógicos dentro de los nodos de una red neuronal permitiendo que procese información simbólica. El problema de este acercamiento es que el entrenamiento clásico de RNAs por el método del descenso del gradiente encuentra un problema al derivar funciones lógicas, ya que éstas no son funciones derivables.

Otra aproximación al paradigma de las FNNs es la representación de reglas difusas mediante una topología neuronal, de tal forma que una red de neuronas puede traducirse fácilmente en una serie de reglas difusas y viceversa. Los antecedentes y consecuentes de las reglas difusas son representados por neuronas distribuidas en diferentes capas de la topología neuronal. La Figura 4.2 muestra una topología típica de FNN y su traducción en reglas difusas. Las capas de la topología neuronal son:

- Capada de entrada: Cada neurona de la capa de entrada representa una de las variables lingüísticas de entrada del FRBS.
- Capa de entrada lingüística: Las neuronas recogidas en esta capa representan las etiquetas lingüísticas de los conjuntos difusos asociados a las variables lingüísticas de entrada representadas por las neuronas de entrada a las que están conectadas.
- Capa de reglas: Cada neurona de esta capa simboliza una regla difusa recogida en la base de reglas del FRBS. Las capas a la izquierda de esta representan los antecedentes de las reglas, y las capas a la derecha el consecuente.
- Capa de salida lingüística: Las neuronas de esta capa son las etiquetas lingüísticas de los conjuntos difusas asociados a las variables lingüísticas de salida representadas por las neuronas de salida a las que están conectadas.

- Capa de salida: Cada neuronas de esta capa representa una variable lingüística de salida del FRBS.

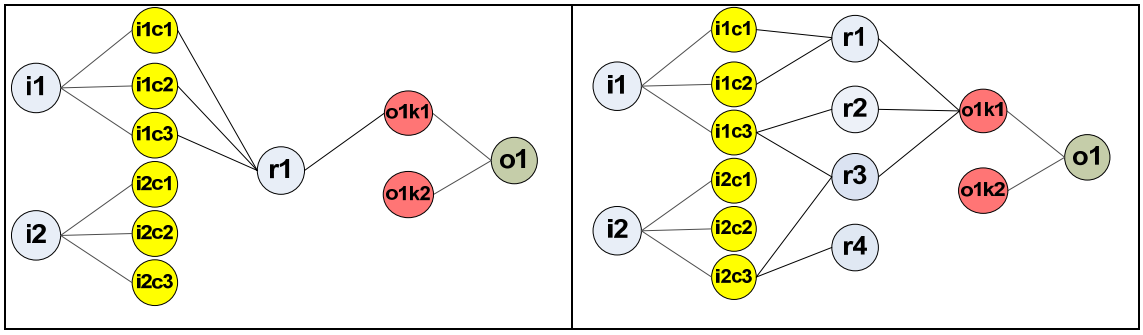


r1	IF i1 IS i1c1 OR i1 IS i1c2 THEN o1 IS o1k1
r2	IF i1 IS i1c2 AND i2 IS i2c2 THEN o1 IS o1k1
r3	IF (i1 IS i1c2 OR i1 IS i1c3) AND i2 IS i2c3 THEN o1 IS o1k1
r4	IF i2 IS i2c3 THEN o1 IS o1k2

Fig. 4.2. Estructura de una FNN.

De la misma forma que una variable lingüística de un FRBS se asocia con varios conjuntos difusos representados por sus etiquetas lingüísticas, las neuronas de la capa de entrada están conectadas con sus correspondientes neuronas de la capa de entrada lingüística. La pertenencia de la neurona i_1 de la capa de entrada a la neurona de la capa de entrada lingüística i_1c_1, viene dada por la función de pertenencia del conjunto difuso representado por i_1c_1 y se expresa en la conexión entre ambas como el peso de la conexión. Las conexiones entre la capa de salida lingüística y la capa de salida siempre tienen peso 1 o 0, representando la presencia o ausencia de dicha conexión. Cada conexión entre una neurona de la capa de entrada lingüística y la neurona r_1 añade un antecedente a la regla representada por r_1, no estando permitida la conexión a r_1 de todas las neuronas de entrada lingüística conectadas a i_1. Las conexiones salientes de r_1 hacia la capa de salida lingüística conforman el consecuente de la regla. Dado que las reglas difusas tienen un único consecuente, sólo se permite una conexión saliente de cada neurona de la capa de reglas, siendo imposible la ausencia de conexiones salientes en una de estas neuronas, ya que una regla no está completa sin su consecuente. Como se

muestra en la Figura 4.2, la estructura de estas FNN permite la implementación de reglas lingüísticas en la forma DNF que incluyen el operador lógico OR.



(a) (b)

Fig.4. 3. Topologías erróneas de FNN.

La Figura 4.3a muestra una topología de FNN errónea, ya que todas las neuronas de la capa de entrada lingüística correspondientes a la neurona de entrada i_1 están conectadas con la neurona de la capa de reglas r_1. La Figura 4.3b ejemplifica otro error, ya que contiene una topología en la cual la neurona r_4 de la capa de reglas no está conectada con ninguna neurona de la capa de salida lingüística.

5. Computación evolutiva

5.1 Conceptos básicos

Desde mediados de los años cincuenta, existen evidencias del uso de ordenadores para, entre otras tareas, avanzar en la comprensión del proceso de la evolución natural [NEUM56], [NEUM58]. Uno de los primeros trabajos que utilizan un proceso evolutivo para la resolución de problemas en computadores es el de Friedberg [FRIE58], [FRIE59]. Estos artículos representan los primeros resultados en aprendizaje automático y describen la utilización de un algoritmo evolutivo para programación automática. El objetivo era encontrar un programa que calculara una función de entrada-salida. También el artículo de Fraser [FRAS57] sirvió de influencia en los primeros trabajos en esta área.

En esta misma época se presentaron los primeros intentos de aplicar evolución simulada a problemas de optimización numérica [BREM62]. Posteriormente, también comenzó el desarrollo de la teoría de los algoritmos evolutivos, demostrando que la relación óptima de mutación para problemas linealmente separables debería tener el valor 1/m, donde m representa el número de bits que codifica cada individuo [BREM65].

Durante este periodo se desarrollaron las ideas de la operación evolutiva (evolutionary operation, EVOP) que utilizaba una técnica evolutiva para el diseño y el análisis de experimentos industriales [BOX57], [BOX69]. Estas ideas no fueron nunca implementadas en un computador, aunque fueron usadas para el desarrollo del método denominado diseño simplex [SPEN62].

Como suele suceder con el comienzo de cualquier nuevo campo de estudio, estos inicios se veían con un alto grado de escepticismo. Sin embargo, a mediados de los sesenta ya están establecidas las bases de lo que hoy son las tres ramas principales de la computación evolutiva (CE). Así los trabajos de Fogel [FOGE66] en San Diego, California, se desarrollaron en el campo de la Programación Genética (PG).

Por otra parte, se desarrollan los algoritmos genéticos por Holland [HOLL69] en la Universidad de Michigan, y las estrategias evolutivas (EE) en la Universidad de Berlín [RECH65].

Durante los siguientes veinticinco años, cada una de estas ramas se desarrolló de una forma bastante independiente, pero con rutas paralelas. Fue en el año 1991 cuando se hizo un esfuerzo por acercar a los distintos investigadores que trabajaban en CE. Para ello tuvo lugar la celebración del Workshop on Paralell Problem Solving from Nature en Dortmund, donde realmente se acuñó el nombre de Computación Evolutiva [SCHW91] para englobar las tres técnicas empleadas para simular los distintos aspectos de la evolución: los algoritmos genéticos, las estrategias evolutivas y la programación genética. Todas ellas tienen en común que utilizan la reproducción, el azar, la competición y la selección de los individuos de la población. Así, en cualquiera de estas técnicas evolutivas están presentes las ideas de la esencia de la evolución y al menos uno de estos cuatro procesos, presuponiendo que, tanto en la naturaleza como en la informática, la evolución es la mejor herramienta de optimización [ATMA76]. Desde un primer momento, las ideas de la CE se orientan hacia cuatro objetivos:

- **Optimización:**

  La evolución en sí es un proceso de optimización [MAYR88] en el que se busca adaptar, lo mejor posible, los individuos al entorno en el que habitan. Darwin [DARW59] estudió, sorprendido, la extrema perfección que ciertos órganos habían alcanzado mediante la evolución, como por ejemplo, los ojos. Sin embargo, optimización no significa perfección. La evolución descubre soluciones funcionales altamente precisas para problemas concretos que se dan en el medio ambiente de un determinado individuo. Debido a esto, ya desde un primer momento, se pensó en su utilización para resolver problemas de optimización en ingeniería. Las técnicas clásicas utilizadas en esta área, como el descenso del gradiente, el "high climbing", o las puramente aleatorias habían sido claramente inadecuadas al enfrentarse a problemas de optimización no lineales, especialmente aquellos que contenían una componente estocástica, temporal o de caos. Todas estas ideas fueron claves para el desarrollo de las EE [RECH94], [SCHW95].

- **Sistemas robustos:**

  Los problemas del mundo real casi nunca son estáticos y los problemas de optimización temporal son cada vez más comunes. Estas circunstancias requieren un cambio en la estrategia que se aplica para resolver el problema. En estos casos, la realimentación sobre el éxito o el fracaso de la estrategia actual es fundamental. Holland [HOLL75], utilizando AAGG, describe un procedimiento que puede hacer evolucionar estrategias, tanto como cadenas codificadas de números o como bases de reglas llamadas sistemas clasificadores. El resultado es un procedimiento robusto que tiene la capacidad de ajustar el rendimiento basándose en la realimentación de la salida del sistema.

- **Inteligencia artificial**:

  Como se puede comprobar, la evolución ha creado especies con una inteligencia cada vez más desarrollada. Por tanto, la forma de conseguir inteligencia artificial, además de las alternativas clásicas que persiguen la emulación de la inteligencia humana, ya sea mediante la emulación de las estructuras biológicas del sistema nervioso o mediante la emulación del comportamiento, tiene una nueva posibilidad: simular la evolución para construir algoritmos predictivos. Esta es la base de las investigaciones en CE [FOGE62], [FOGE66].

- **Vida artificial:**

  En el campo de la biología, más que utilizar la evolución como una herramienta, se intenta capturar la esencia de la propia evolución en una simulación por computador y usar entonces esta simulación para conseguir una nueva información sobre la física de los procesos de la evolución natural [RAY91]. El éxito en este campo abre la posibilidad de estudiar sistemas biológicos alternativos de cómo puede ser la vida en otros lugares y averiguar qué características pueden tener en común con la de La Tierra [LANG87]. Aunque cada modelo es incompleto y las características que tiene la vida en otros lugares del universo es pura especulación, las simulaciones de vida por ordenador, denominadas genéricamente vida artificial, ya han conseguido generar algunos patrones que se corresponden con fenómenos que normalmente ocurren en la vida en La Tierra.

Una vez vistas las distintas ramas de aplicación de la CE, la cuestión fundamental que se plantea ahora es por qué simular los procesos de la evolución. Una respuesta podría ser que en estos momentos no se dispone de mejores alternativas. Normalmente, no se pueden utilizar eficientemente las técnicas clásicas de optimización para encontrar un máximo global en una función si está rodeado de máximos locales. Las técnicas desarrolladas hasta ahora para emular la inteligencia humana no son suficientemente adaptables a los dominios cambiantes, ya que no son suficientemente capaces de predecir ni de anticipar sus acciones. Tampoco se puede visitar un nuevo mundo, enviarle los compuestos químicos primigenios que dan lugar a la vida y esperar millones de años para ver qué tipo de vida se desarrolla allí. En contraste, los métodos de computación evolutiva ofrecen posibles soluciones a estos problemas basándose en la utilización dirigida del azar y la incertidumbre.

De acuerdo con el teorema de no-free-lunch (NFL) [WOLP96], no puede existir ningún algoritmo que resuelva todos los problemas (por ejemplo de optimización) que sea, en general, superior a cualquier otro. Por tanto, en principio, la cuestión de si los algoritmos evolutivos son superiores o inferiores a cualquier otra aproximación no tiene sentido. Sin embargo, es posible afirmar que los algoritmos evolutivos se comportan mejor que otros métodos con respecto a problemas específicos y, como consecuencia, se comportarán peor con respecto a otras clases de problemas.

El teorema NFL puede corroborar que, en el caso de los algoritmos evolutivos frente a otros métodos clásicos de optimización, son más eficientes resolviendo problemas discontinuos, no diferenciables, multimodales, con ruido y cualquier tipo de superficies de búsqueda no convencionales. Por el contrario, su efectividad disminuye al afrontar problemas más simples para los que se han desarrollado algoritmos específicos en su resolución.

Por tanto, en principio, y mientras no se desarrolle un método específico de resolución del problema que tenga en cuenta ciertas propiedades del mismo que lo haga más eficiente, parecen claras las ventajas de la utilización de algoritmos no especializados y robustos como los algoritmos evolutivos.

5.1.1 Bases de la computación evolutiva

La base más comúnmente aceptada de teorías sobre la evolución es el paradigma neo-Darwiniano. En esta teoría se afirma que la mayor parte de la historia de la vida puede ser completamente justificada mediante procesos físicos y mediante poblaciones y especies [HOFF89]. Estos procesos son: la reproducción, la mutación, la competición y la selección. La reproducción es una propiedad común de las especies. Hay que tener en cuenta que, normalmente, la capacidad reproductiva de las especies es enorme y el número de individuos se incrementaría exponencialmente si todos sus individuos se pudieran reproducir con éxito simultáneamente. La reproducción supone la transferencia del código genético de cada individuo a su descendencia. La mutación sucede debido a que los errores en la replicación durante el proceso de transferencia de información genética son inevitables, además de ser necesario incluir variabilidad en las especies. La competición es una consecuencia del crecimiento de la población dentro de un espacio físico finito donde no hay espacio o energía para todos. La selección es el resultado de la reproducción de las especies en un medio de competencia. Gracias a ella, sobrevivían los mejores individuos, esto es, los más adaptados al medio.

Los individuos y las especies pueden ser vistos como una dualidad de su código genético, el genotipo, y su forma de plasmarse con respecto al mundo, el fenotipo. El genotipo ofrece la posibilidad de almacenar la experiencia acumulada, adquirida de la información histórica. Desgraciadamente, el resultado de la variación genética es impredecible debido a los efectos combinados de la pleitropía y la poligénesis, [MAYR88]. La pleitropía implica que un único gen afecta simultáneamente a varias características del fenotipo, mientras que la poligénesis significa que cada característica del fenotipo viene determinada por la interacción de varios genes. No se conoce ninguna relación, uno a uno, entre genes y características en los sistemas evolutivos naturales. Debido a esto, el fenotipo cambia siguiendo una función compleja, no lineal, de la interacción entre las estructuras genéticas y las condiciones medioambientales. De esta forma, códigos genéticos muy distintos pueden tener comportamientos equivalentes, igual que programas diferentes pueden conseguir resultados similares.

Dentro del proceso de evolución, hay que tener en cuenta que la selección actúa sólo sobre los comportamientos externos de los individuos y las especies [MAYR88] y no sobre el genotipo que representan. La selección viene dada por la adaptación de los individuos al entorno, y esta adaptación se puede ver como un espacio con una topografía adaptativa dependiente de las condiciones del entorno. Una población de genotipos genera sus respectivos fenotipos que, por sus características de adaptación al medio, están situados en un cierto lugar del mapa de adaptación. Cada pico de este mapa se corresponde con una zona de fenotipos optimizados. De una forma probabilística, la evolución acerca a los individuos a estos picos, mientras que la selección elimina las variantes fenotípicamente peores.

Visto de esta forma, la evolución es claramente un proceso de optimización en resolución de problemas. La selección conduce los fenotipos tan cerca como es posible del óptimo para la supervivencia con respecto a unas condiciones iniciales y unas restricciones del entorno. Sin embargo, el entorno está continuamente cambiando y, por tanto, las diferentes especies constantemente evolucionan hacia nuevos óptimos. Debido a esto, no se puede decir que ningún organismo esté perfectamente adaptado a su entorno.

Según todo lo visto anteriormente, se resumen a continuación las características más importantes del paradigma neo-Darwiniano:

- El individuo es el principal objetivo de la evolución.
- La variación genética es un fenómeno de probabilidad. Los fenómenos estocásticos juegan un papel importante en la evolución.
- La variabilidad de genotipo se basa, en su mayor parte, en un proceso de recombinación y, sólo en menor medida, en la mutación.
- La evolución gradual puede incluir discontinuidades en el fenotipo.
- No todos los cambios en el fenotipo tienen que ser consecuencia de la selección natural.
- La evolución es un cambio en la adaptación y la diversidad, no solamente un cambio en la frecuencia de los genes.
- La selección es probabilística, no determinista.

5.1.2 Características de la computación evolutiva

Todas las técnicas de la CE se basan en el modelo de la evolución natural, por lo que todos comparten un conjunto de características comunes. A continuación se reseñan las más importantes:

- Se utiliza una estrategia de aprendizaje colaborativo a partir de un conjunto de individuos. Normalmente, cada individuo representa o codifica un punto del espacio de búsqueda de soluciones en un problema dado. Cada individuo incorpora información adicional que permite llegar a la solución final del problema.

- La descendencia, obtenida a partir de los individuos de la población, se genera de forma pseudo-aleatoria mediante procesos de cruce y recombinación donde se intercambia información entre dos o más individuos actualmente existentes. La mutación es un proceso de auto-replicación errónea de los individuos que produce pequeños cambios en los mismos.

- Mediante la evaluación de los individuos, se consigue una medida de la adaptación de cada uno de ellos a su entorno. De acuerdo con esta medida de adaptación, el proceso de selección favorece más a los individuos mejor adaptados, que serán por tanto los que se reproduzcan más frecuentemente.

5.2 Algoritmos Genéticos

5.2.1 Introducción

Los Algoritmos Genéticos (AAGG) son un tipo de algoritmos evolutivos usados para resolver problemas de búsqueda y optimización [SCHW91]. Se basan en la imitación del proceso evolutivo que se produce en la naturaleza para resolver problemas de adaptación al medio. De esta manera, son capaces de hacer evolucionar los individuos de una población hacia soluciones mundo real siempre que se encuentre una

codificación adecuada del problema, siendo ésta una de las claves principales de éxito cuando se aplica este tipo de técnicas.

Los AAGG presentan, por tanto, una analogía directa con el proceso de evolución por selección natural descrito por Darwin [DARW59]. A cada individuo se le asigna un valor dependiendo de lo buena que sea la solución que aporta al problema. A los individuos con mayor adaptación, esto es, a los que mejores soluciones representan, se les da mayor oportunidad de reproducirse, cruzándolos con otros individuos de la población que probablemente representan también buenas soluciones al problema. Así se produce la descendencia cuyos individuos comparten características de cada uno de los padres. Se van produciendo por tanto, nuevas generaciones con una proporción mayor de individuos mejor adaptados. De esta forma, con el paso de las generaciones, las poblaciones constan de individuos que aportan mejores soluciones que, a priori, pueden considerarse como peores, pero que al recombinarse pueden dar lugar a individuos muy bien adaptados. Si el AG ha sido bien diseñado, la población convergerá hacia la solución óptima del problema.

La potencia de los AAGG se debe a que la técnica que usan es robusta y puede tratar con éxito un gran número de tipos de problemas, incluyendo y destacando aquellos que son difícilmente solucionables utilizando otro tipo de métodos clásicos. Los AAGG no garantizan la localización exacta de la solución óptima del problema, pero proporcionan una solución alternativa aceptablemente buena.

5.2.2 Características y funcionamiento genérico

Los AAGG comienzan con una muestra elegida de manera aleatoria en el espacio de soluciones, para ir transformándola paso a paso hasta llegar a un estado estacionario, en el cual la muestra, también llamada población, está constituida por las soluciones del problema. Un AG no actúa, en general, directamente sobre el espacio de soluciones, sino sobre una codificación de éste, haciendo interaccionar entre sí y operando sobre las cadenas resultantes de dicha codificación. Esta acción tiene como consecuencia la progresiva transformación del conjunto de elementos evaluados, hasta llegar al estado final. Para ello, los algoritmos genéticos se valen de la acción de los llamados

operadores genéticos, tales como selección, cruce y mutación, a la que es sometida la población. Cada operador juega un papel diferente en la evaluación de la población. Así, la selección puede ser entendida como una competición, mientras que los otros operadores, tales como el cruce y la mutación son capaces de crear nuevos individuos a partir de los existentes en la población. La figura 3.1 muestra el funcionamiento genérico de los AAGG.

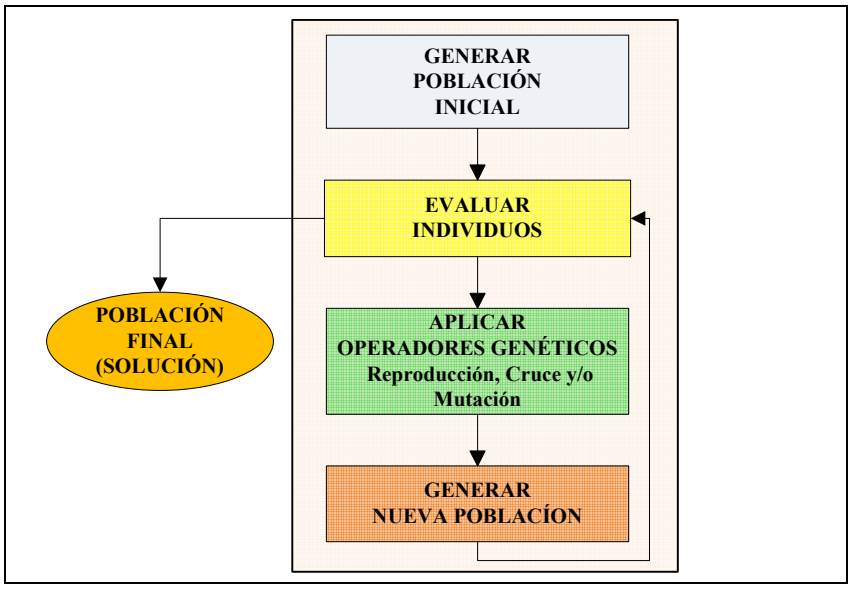


Fig. 5.1: Esquema general de funcionamiento de los algoritmos genéticos

5.2.3 Codificación del problema

Cuando se trabaja con AAGG, se precisa algún método de codificación de los puntos del espacio de búsqueda de la función con la que se está trabajando. Esta es una de las características más interesantes y peculiares del método, ya que con ello cada punto se puede tratar como una cadena, en función de la codificación utilizada. Por tanto, una vez aplicada la codificación, el dominio de la función a optimizar queda transformado en un conjunto acotado de valores representados por cadenas, independientemente de su origen.

Habitualmente, la codificación utilizada suele ser completa. Es decir, si se utiliza un alfabeto A de cardinal finito m, y cadenas de longitud l, cualquiera de las m^l posibles cadenas corresponde a la codificación de un único punto del dominio de la función [BARR91]. Sin embargo, dada la discretización establecida por la codificación, la correspondencia anterior no es biunívoca entre el dominio de la función objetivo y las diferentes cadenas.

Los individuos que forman una población se representan mediante un conjunto de parámetros denominados genes. A cada uno de los posibles valores que puede tomar un determinado gen, se le denomina alelo. El número de alelos posibles de un gen coincide con el cardinal del alfabeto utilizado: m. Si la codificación es binaria entonces existen dos posibles alelos por cada gen que son el valor cero y el valor uno. Por último, los genes a su vez se agrupan para formar cromosomas. En la literatura existente, se suele identificar individuo con cromosoma aunque, en general, en la naturaleza, un individuo consta de varios cromosomas.

5.2.4 La función de evaluación

La función de evaluación, también denominada función objetivo, se diseña de forma específica para cada problema que se quiere resolver [DORA99]. Dado un individuo, la función de evaluación devuelve un valor de ajuste o de adaptación proporcional a la habilidad de dicho individuo para resolver el problema en cuestión.

La elección de una función objetivo adecuada es, junto con la codificación de los individuos, uno de los aspectos cruciales en el comportamiento de los AAGG. Una característica que se considera interesante a la hora de construir funciones objetivo es que se cumpla que dos individuos que se encuentran cercanos en el espacio de búsqueda tengan valores parecidos de la función. También es deseable que la función objetivo no presente una cantidad excesiva de óptimos locales.

5.2.5 Convergencia

Si el algoritmo genético ha sido planteado correctamente, la población evoluciona a través de las generaciones de tal forma que la adaptación del mejor y la media de todos los individuos en cada generación se incrementa hacia el óptimo global. Este proceso de incremento uniforme del valor de adaptación es lo que se denomina convergencia. Se dice que un gen converge cuando el 95% de la población comparte el mismo valor para ese determinado gen [DEJO75], o bien que todos los valores que toma ese gen en todos los individuos de la población sufren variaciones de no más del 5% para el caso de alfabetos finitos. Asimismo, se dice que la población converge cuando todos los genes han convergido.

5.3 Programación Genética

5.3.1 Características

La Programación Genética (PG) es una extensión de los Algoritmos Genéticos (AAGG) en la que la población está formada por programas de computación representados en forma de árboles sintácticos. Ambos enfoques son muy similares en la aplicación del modelo evolutivo; sin embargo, en los AAGG se establece a priori un esquema adecuado para la representación de los individuos en el problema (generalmente una cadena de caracteres de longitud fija), y se opera sobre esa estructura predefinida para llegar a una solución aceptable. Mientras, en la PG, se escogen los elementos que conforman los individuos, es decir, las acciones de los programas (funciones) y los datos sobre los que actúan los programas (terminales), y se opera con ellos hasta encontrar una estructura que represente una solución óptima al problema.

La PG busca construir programas de computación sin que ellos sean diseñados y programados expresamente por un humano. Puede decirse que es una técnica de optimización cuyo espacio de búsqueda son todos los posibles programas que solucionan un problema determinado. Debe aclararse que la creación de esos programas es completamente aleatoria, es decir, el computador no ha sido explícitamente programado para encontrar una solución predeterminada, sino que a esta solución se

llega partiendo de un estado inicial dado, aplicando mecanismos de selección a las estructuras intermedias que se van encontrando [KOZA92].

El conjunto de las posibles estructuras lo integran todas las combinaciones de funciones que se pueden componer recursivamente a partir de un alfabeto, de cardinal finito m, compuesto por un conjunto de N_{func} funciones ($F=\{f_1, f_2, \ldots, f_{Nfunc}\}$) y un conjunto formado por N_{term} terminales ($T=\{a_1, a_2, \ldots, a_{Nterm}\}$). Cada función f_i del conjunto F, trabaja con un número específico $z(f_i)$ de argumentos.

Las funciones pueden ser:

- Operadores aritméticos (+, -, *, …)
- Funciones matemáticas (sin, cos, exp, log, …)
- Operadores booleanos (AND, OR, NOT, …)
- Operadores condicionales(if-then-else)
- Funciones de iteración (Do-until)
- Funciones recursivas
- Funciones específicas del dominio que deben ser definidas

Los terminales pueden ser variables (pudiendo representar entradas, sensores, detectores o estados de un sistema) o constantes [KOZA92]. Los genes se corresponden con los nodos de los árboles, y los alelos con las diferentes funciones y terminales, que dependen del dominio del problema.

Una vez generada aleatoriamente la población inicial, el funcionamiento de la PG es muy similar al de los AAGG, aplicando los operadores genéticos a los individuos hasta llegar a la solución deseada para el problema en cuestión.

5.3.2 Operadores genéticos

Entre los operadores genéticos, los considerados fundamentales por la mayoría de los estudiosos del tema [HOLL75], [GOLD89], son los de selección, mutación y cruce. Cada uno de ellos desarrolla una labor específica, bien diferenciada de las de los demás.

- **Operadores de selección:**

  El operador de selección se encarga de seleccionar a los individuos destinados a tener descendencia agrupándolos en un lugar intermedio denominado lugar de apareamiento (matting pool). Los individuos mejor adaptados son aquellos que tienen mayor probabilidad de ser escogidos, por lo que la mayoría de los operadores de selección se basan en el valor proporcionado por la función objetivo para cada individuo. Todos los operadores de selección ofrecen un rendimiento similar si se utilizan los parámetros adecuados [GOLD91].

  El método de selección más utilizado actualmente es el método del torneo [BRIN91]. Éste consiste en dividir a una población que consta de N individuos en conjuntos de K individuos. De esta forma se dirimen N/K torneos, y cada uno de los ganadores (es decir, aquel individuo mejor adaptado) de cada torneo es seleccionado para la reproducción. Típicamente, suele tomarse K=2 (torneo binario), realizándose N/2 torneos y por tanto N/4 cruces.
  Otros métodos de selección utilizados habitualmente son el método de la ruleta [BAKE87], el método de truncamiento [CROW70], utilizado cuando se trabaja con poblaciones con un número elevado de individuos, y el método generacional [DEJO75].

- **Operadores de cruce:**

  El operador de cruce se considera el más importante de los operadores genéticos. Su función consiste en cruzar cada pareja de padres que se encuentran en el lugar de apareamiento para obtener su descendencia y así continuar con la evolución de la población. De este modo, cuanto mejor sea el operador de cruce elegido, más rápidamente converge la población hacia la solución del problema. Su funcionamiento es sencillo: se trata de elegir un nodo de cruce o lugar de cruce para cada uno de los dos padres y a continuación intercambiar los subárboles cuyas raíces son esos nodos de cruce. De esta forma, se obtienen dos nuevos individuos que, dependiendo del grado de adaptación alcanzado y de la política de reemplazo escogida, pueden pasar a formar parte (o no) de la nueva población.

- **Operadores de mutación:**

  El operador de mutación, actúa sobre un determinado individuo con una probabilidad p_m. Cuando un individuo resulta afectado por el operador, se selecciona aleatoriamente un lugar de mutación. Para ello, todos los lugares son equiprobables. El subárbol cuya raíz es el nodo elegido para la mutación es reemplazado por un subárbol cuyo símbolo del nodo raíz coincide con el símbolo del nodo elegido para realizar la mutación.

  El operador de mutación encuentra su más amplia justificación como método para obtener nuevos puntos en los que recomenzar la búsqueda, evitando de esta forma la pérdida de diversidad genética en la población, lo cual tiene implicaciones directas en el proceso de convergencia del algoritmo. Tanto es así que, según ciertos estudios, el operador de mutación va ganando en importancia a medida que la población de individuos va convergiendo [DAVI91].

- **Reemplazo de individuos:**

  Una vez obtenidos los individuos descendientes de una determinada población tras la aplicación de los diferentes operadores, es necesario formar una nueva población (la nueva generación) de λ individuos a partir de los λ individuos de la población de partida y los nuevos individuos obtenidos. El concepto de reemplazo suele ir ligado con el de tasa de reemplazo generacional t_{tg} que es el porcentaje de hijos generados con respecto al tamaño de la población inicial.

  Inicialmente, el proceso de reemplazo se efectuaba de uno en uno, $t_{tg} = 1/\lambda$, por lo que se generaba un único individuo descendiente que sustituía a otro de la población inicial [HOLL75].

  De Jong aplicó el concepto de tasa de reemplazo generacional con el objetivo de efectuar un solapamiento controlado entre padres e hijos [DEJO75]. En este trabajo, una proporción t_{tg} de la población es seleccionada para ser cruzada. Los hijos resultantes reemplazarán a los miembros de la población anterior que peor adaptados se encuentren. Este tipo de algoritmos genéticos, donde únicamente se

sustituyen unos pocos individuos, se conocen bajo el nombre de SSGA (Steady-State Replacement Genetic Algorithms), llegando en algunas implementaciones a sustituir únicamente dos.

5.4 Programación Genética guiada por Gramáticas

5.4.1 Características

La Programación Genética Guiada por Gramáticas (PGGG) surge como una extensión a los sistemas tradicionales de PG [WHIG95]. La diferencia estriba en que se utilizan gramáticas de contexto libre (del inglés "context free gramar" o CFG) para establecer la definición formal de las restricciones sintácticas del problema. El hecho de utilizar este tipo de formalismos sintácticos ayuda a resolver el denominado problema del cierre (del inglés "closure") [KOZA92], que hace referencia a que el cruce de dos individuos válidos genere también descendientes válidos, puesto que la generación de individuos inválidos entorpece mucho la velocidad de convergencia del sistema. La PGGG ha demostrado ser una aproximación con un gran rendimiento en problemas con dominios estructurados [WHIG95], [WHIG96] y es considerada como una de las áreas más prometedoras dentro de la investigación de la PG. Una de las aplicaciones más novedosas de la PGGG es su utilización en la generación automática de topologías neuronales [COUC07].

En esencia, la PGGG utiliza los mismos componentes y operadores genéticos que la PG tradicional. Sin embargo existen diferencias significativas entre ambos sistemas. La primera diferencia fundamental aparece a la hora de generar aleatoriamente la población inicial: los individuos que conforman la población deben ser palabras aceptadas por el lenguaje definido por la gramática libre de contexto en cuestión. La utilización de los métodos anteriores para inicializar la población en los sistemas de PG no son válidos para la PGGG dado que los individuos se generan aleatoriamente a partir del conjunto de funciones y de terminales, esto conlleva la generación de individuos no válidos que impiden la convergencia hacia la solución. Otra diferencia fundamental aparece a la hora de diseñar el operador de cruce, puesto que los operadores tradicionales de cruce para PG no aseguran la generación de descendencia válida sintácticamente de acuerdo a

la gramática. Por último, el operador de mutación, una vez elegido el lugar de mutación, debe utilizar las reglas de producción de la gramática para asegurar la mutación del individuo por otro igualmente válido.

5.4.2 Generación de la población inicial

- **Fuerza bruta:**

  El algoritmo de fuerza bruta [KARP87] describe un estilo de programación en el que el programador confía en la capacidad de cómputo del ordenador en vez de usar su inteligencia para simplificar el problema. Su ámbito de uso es enorme, pudiéndose aplicar a una gran variedad de problemas. En particular, este algoritmo puede ser utilizado a la hora de generar aleatoriamente la población inicial del sistema de PGGG. Dada la gramática libre de contexto que representa el problema en cuestión, el algoritmo utiliza las producciones que conforman la gramática para ir formando cada uno de los individuos que componen la población. Los individuos generados son válidos sintácticamente, puesto que se generan siguiendo las producciones de la gramática.

  El algoritmo comienza escogiendo alguna producción de la gramática que parta del axioma. Posteriormente, para cada símbolo no terminal que aparece se escoge (de nuevo de forma aleatoria) alguna producción que contenga en la parte izquierda a dicho símbolo no terminal, y así recursivamente hasta que se alcancen únicamente símbolos terminales.

- **Algoritmo GBPG:**

  El algoritmo de generación aleatoria de individuos (GBPG, Grammar Based Population Generator) [GARC07] tiene como objetivo establecer la población inicial de un sistema de Programación Genética Guiada por Gramáticas. La característica fundamental que presenta este algoritmo reside en que a la hora de generar los individuos, no se eligen las producciones que lo conforman de manera totalmente aleatoria, sino que se eligen producciones que aseguran en todo momento que el individuo pertenece al lenguaje generado por la gramática y que la profundidad de su árbol de derivación no va a superar una cota

previamente establecida. Esto proporciona un ahorro computacional importante en términos de tiempo y memoria dado que ningún individuo es descartado a la hora de formar parte de la población inicial.

El algoritmo recibe dos parámetros: el número de individuos (N) que componen la población y la profundidad máxima permitida (D) de los árboles de derivación.

El proceso de generación aleatoria de la población consta de tres pasos:

1. Se anota la longitud para cada producción de la gramática. Para ello se tienen en cuenta las siguientes definiciones:

   - La longitud de un símbolo no terminal A es el mínimo de las longitudes de todas sus producciones, y se denota por L(A).

   - La longitud de un símbolo terminal a es 0 porque no deriva a nada, denotándose por L(a) = 0.

   - La longitud de una producción A::= α es el resultado de sumar uno al máximo de las longitudes de los símbolos que componen la parte derecha, y se denota por L(A::= α).

   - Las producciones que generan sólo símbolos terminales tienen longitud 1, denotándose por L(A:=a) = 1, $\forall A \in \Sigma_N$ y $\forall a \in \Sigma^*_T$.

2. Se calcula la longitud del axioma (mínimo de las longitudes de todas aquellas producciones cuyo antecedente es el axioma). Esta longitud determina la profundidad mínima permitida (d) para un individuo válido de esa gramática.

3. Repetir N (número de individuos de la población) veces:

   a. Se escoge aleatoriamente un valor comprendido entre la profundidad mínima permitida (d) y la profundidad máxima permitida (D). Este valor corresponde a la profundidad máxima elegida (p) para el presente individuo. El algoritmo garantiza la generación de un individuo cuya profundidad está comprendida entre d y p.

   b. Se etiqueta el axioma como el actual no terminal A y se asigna el valor 0 a la profundidad actual (PA).

   c. Se selecciona aleatoriamente una producción de la forma A ::= α, ($\alpha \in \{\Sigma_N \cup \Sigma_T\}^*$) que cumpla: $PA + L(A ::= \alpha) \leq p$.

   d. Por cada no terminal $B \in \alpha$, se anota B como el actual no terminal A y se repiten los pasos 3.3 y 3.4, incrementando en una unidad el valor de PA.

   La figura 5.2 muestra el proceso de generación del individuo 6 + 4 = 7 perteneciente a la gramática de igualdades aritméticas con sumas y restas, mediante el algoritmo propuesto.

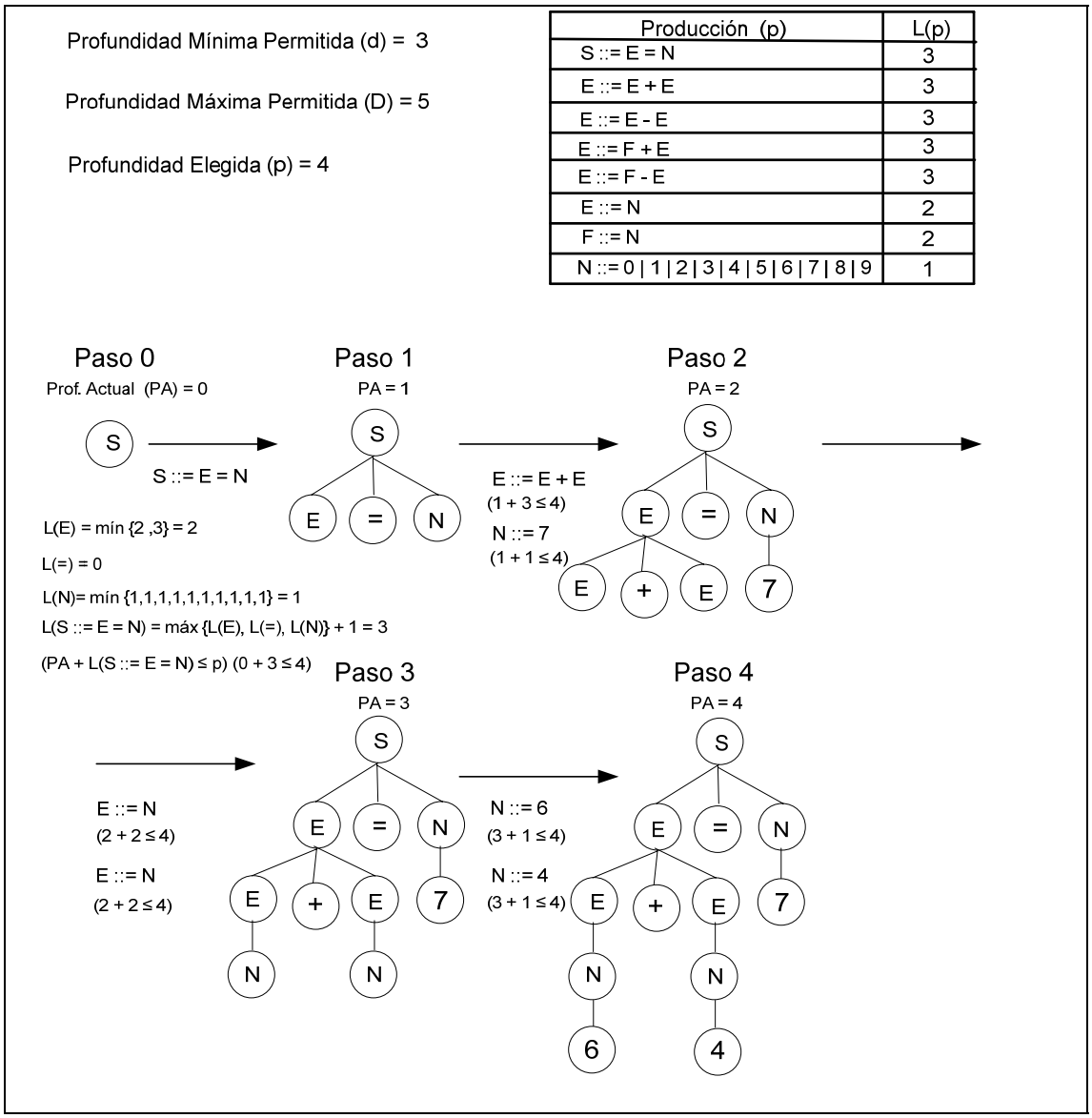


Fig. 5.2: Generación del individuo 6 + 4 = 7 mediante el algoritmo de generación aleatoria de poblaciones

5.4.3 Operadores de cruce

- **Whigham (WX):**

  Este operador propuesto por Whigham [WHIG95] constituye el operador de cruce más ampliamente utilizado en PGGG. Su buen funcionamiento para este tipo de sistemas ha hecho que se convierta en el estándar de facto y que aún en nuestros días siga completamente vigente [RODR02], [HUSS03], [GROS04]. Ciertos investigadores estudian el interés de utilizar este operador en problemas definidos mediante gramáticas ambiguas [HOAI02].

La utilización de una gramática libre de contexto hace necesario definir un operador de cruce que garantice la generación de individuos válidos, es decir, palabras que pertenezcan al lenguaje generado por tal gramática.

El algoritmo consta de los siguientes pasos: en primer lugar, dado que todos los símbolos terminales tienen al menos un símbolo no terminal por encima de ellos, entonces sin pérdida de generalidad puede restringirse que los nodos de cruce se localicen exclusivamente sobre nodos no terminales. Posteriormente, se escoge aleatoriamente un nodo no terminal que pertenezca al primer padre y tras ello se escoge en el segundo padre un nodo no terminal etiquetado con el mismo símbolo no terminal que en el primer punto de cruce escogido.

De esta forma se asegura la generación de individuos que pertenecen al lenguaje generado por esa gramática, dado que se cruzan nodos con el mismo símbolo. Si no existe en el segundo padre un nodo con el mismo símbolo no terminal que en el primer nodo de cruce, entonces se vuelve a elegir otro nodo de cruce de entre los símbolos no terminales del primer padre y se repite la operación hasta que se pueda efectuar el cruce.

- **Operador de cruce gramatical (GBX)**

  El operador de cruce gramatical (GBX, Grammar Based Crossover) [MANR05] es un operador de propósito general para la resolución de problemas con el paradigma de la Programación Genética Guiada por Gramáticas. GBX presenta tres características reseñables: evita el crecimiento desmesurado del tamaño de los árboles de derivación que representan a los individuos [COUC06], fenómeno conocido como explosión de código (del inglés “code bloat”) [PANA04], proporciona un equilibrio adecuado entre la capacidad de exploración del espacio de búsqueda y la capacidad de explotación, preservando el contexto en el cual los subárboles aparecen en los árboles padre; y finalmente es capaz de aprovechar la propiedad de las gramáticas ambiguas consistente en la existencia de más de un árbol de derivación distinto para una misma palabra. La unión de todas estas características proporciona a GBX una elevada velocidad de convergencia y una reducida probabilidad de caída en óptimos locales.

El operador de cruce gramatical consta de diez pasos:

1. Se crea un conjunto NT compuesto por los nodos no terminales del primer padre (salvo la raíz). Un nodo queda denotado mediante una tupla N = (Z, coord), donde Z es el no terminal que representa el nodo y coord es la coordenada de dicho nodo en el árbol, utilizando la notación del operador de cruce SCPC.

2.

   a. Si NT ≠ Φ, entonces se elige un elemento de dicho conjunto de manera aleatoria, el cual se denomina nodo de cruce o lugar de cruce (NC1).

   b. Si NT = Φ, entonces el cruce entre los individuos no es posible y se elige la raíz como nodo de cruce para ambos árboles, dando lugar a descendientes idénticos a los padres.

3. Se busca el nodo padre del nodo de cruce elegido anteriormente. Éste siempre es un símbolo no terminal, puesto que se trabaja con una gramática libre de contexto. Éste símbolo aparece como antecedente de una o varias reglas de producción de la gramática. A continuación, se almacenan los consecuentes de todas estas producciones en un array R.

4. Se denomina derivación principal (A :: = C), a la derivación que produce el nodo padre del nodo NC1. Se definen asimismo la longitud de la derivación principal (l), como el número de símbolos terminales y no terminales incluidos en el consecuente de la derivación principal, y la posición (p), que ocupa el nodo de cruce en la derivación principal. Se calcula y almacena esta tupla T= (l,p,C).

5. Se eliminan del array R todos aquellos consecuentes cuya longitud sea diferente a la longitud de la derivación principal (l).

6. Para cada elemento de R, se comparan todos los símbolos con los del consecuente de la derivación principal excepto aquél que ocupa la misma posición (p) que el nodo de cruce. Posteriormente, se eliminan de R todos aquellos consecuentes en los que se ha detectado alguna diferencia.

7. Se calcula el conjunto X, formado por todos aquellos símbolos en los consecuentes de R, que están en la misma posición (p) que el nodo de cruce.

8.

   a. Si $X \neq \Phi$, se elige aleatoriamente algún símbolo (SE, símbolo elegido) que pertenezca al conjunto X. Se almacenan todos los nodos candidatos del segundo padre cuyo símbolo coincida con SE. Dichos nodos conforman el conjunto NC (nodos candidatos).

   b. Si $X = \Phi$, entonces no existen nodos de cruce del segundo padre que satisfacen los requisitos de cruce para los símbolos del conjunto X, por lo que se elimina el nodo de cruce del primer padre (NC1) del conjunto NT y se continúa en el paso 2.

9.

a. Si $NC \neq \Phi$, se escoge aleatoriamente un nodo de cruce (NC2) del conjunto de nodos candidatos NC del segundo padre.

b. Si $NC = \Phi$, entonces se elimina SE del conjunto X y se va al paso 8.

10. Se calcula la profundidad a la que se encuentra el nodo de cruce del primer padre (NC1) y se le suma la longitud del subárbol cuya raíz es el nodo de cruce del segundo padre (NC2), la profundidad obtenida se denomina P1. Se calcula la profundidad a la que se encuentra el nodo de cruce del segundo padre (NC2) y se le suma la longitud del subárbol cuya raíz es el nodo de cruce del primer padre (NC1), la profundidad obtenida se denomina P2.

a. Si P1 ó P2 superan el valor de la profundidad máxima (D) definida para un individuo, entonces se elimina NC2 del conjunto NC y se va al paso 9.

b. En caso contrario, se obtienen los símbolos no terminales de NC1 y NC2 y se comparan.

- Si ambos símbolos no terminales coinciden, entonces los dos nuevos descendientes se obtienen intercambiando los dos subárboles cuyas raíces coinciden con los nodos de cruce previamente calculados (NC1 y NC2).
  En caso contrario, se calcula la derivación generada por el nodo padre de NC2 y se obtiene la posición (p2) del símbolo del NC2 en la derivación, sustituyendo en ésta el símbolo de la posición p2 por el símbolo del NC1.
- Si la derivación obtenida coincide con algún consecuente de las reglas de producción de la gramática, entonces los dos nuevos descendientes se obtienen intercambiando los dos subárboles cuyas raíces coinciden con los nodos de cruce previamente calculados (NC1 y NC2).
- En caso contrario, se elimina NC2 del conjunto NC y se va al paso 9.

La figura 5.3 muestra el resultado de aplicar este método de cruce.

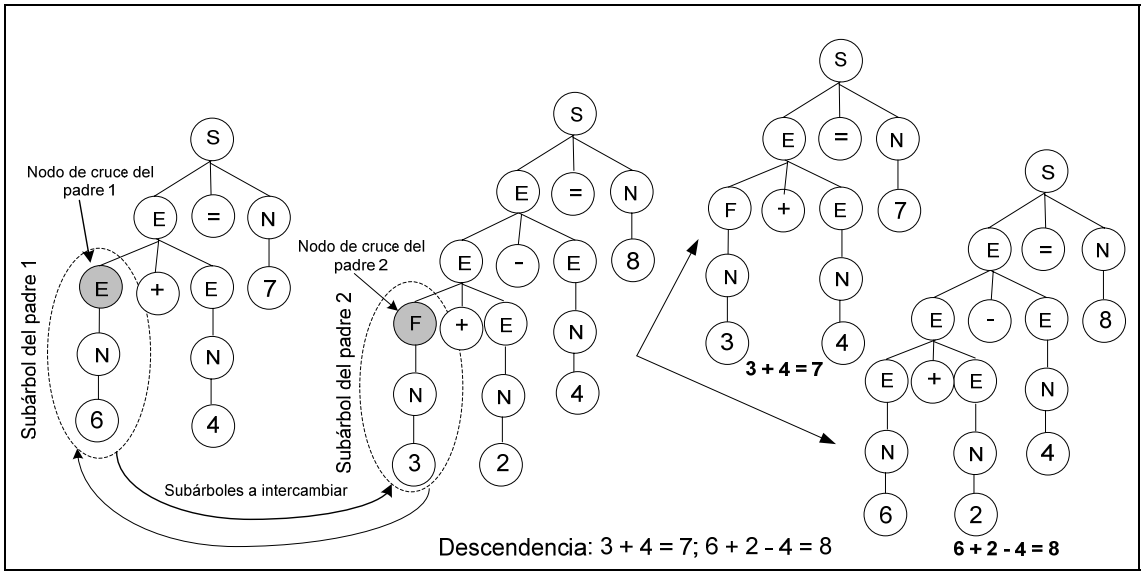


Fig. 5.3: Ejemplo de funcionamiento del operador de cruce GBX

Las características fundamentales de este operador de cruce se resumen en los siguientes puntos:

- Los pasos del algoritmo aseguran que la descendencia obtenida como resultado de aplicar el operador de cruce siempre da lugar a individuos válidos, es decir, sintácticamente correctos de acuerdo a la gramática dada. Este factor es sumamente importante dado que se favorece en gran medida el proceso de evolución, ya que todos los individuos generados son posibles soluciones al problema.

- GBX tiene en cuenta todos los posibles nodos del segundo padre que pueden generar individuos válidos, una vez elegido el nodo de cruce para el primer padre.

- Resulta sencillo incorporar al algoritmo un mecanismo de control eficiente que limita el crecimiento en exceso de los árboles, descartando todos aquellos posibles nodos de cruce que generan árboles descendientes de una profundidad superior a una profundidad máxima permitida (D) previamente establecida. Esto conlleva un ahorro de recursos computacionales (tiempo y memoria), dado que los individuos generados son de menor profundidad.

6. Planteamiento del problema

6.1 Introducción

Los sistemas basados en reglas, tanto en su aproximación clásica como difusa, se han utilizado con éxito en la resolución de problemas de diversas áreas de aplicación: monitorización, diagnóstico, control, clasificación, modelado, etc… El proceso de construcción de un RBS consta de dos fases diferenciadas: el diseño del motor de inferencia y la generación de la base de conocimiento.

El diseño del motor de inferencia no presenta grandes dificultades ya que únicamente supone la elección del sistema de inferencia que mejor se adecúa a las necesidades del RBS que se pretende desarollar. Atendiendo a las características de éste (la inclusión o no de reglas difusas, el tipo de encadenamiento, etc…), las opciones a elegir varían, pero en ningún caso supone un obstáculo en el proceso de construcción del RBS.

La mayor dificultad del proceso de construcción de un RBS radica en la generación de la base de conocimiento, compuesta por la base de hechos y la base de reglas. Existen dos clases de métodos para crear ambas bases: manuales y automáticos. Los manuales implican la participación de un experto, del cual se educe el conocimiento necesario para construir una base de reglas y una base de hechos. Los métodos puramente automáticos comprenden técnicas de minería de datos para elaborar la base de hechos y algoritmos de aprendizaje para generar la base de reglas, ambos sin mediación de expertos. El mayor inconveniente de las técnicas manuales es que son muy costosas dada su dependencia del experto, conformando el gran cuello de botella de la ingeniería del conocimiento. Además dan lugar a bases cuyo conocimiento es subjetivo, debido a que ha sido educido de un ser humano. Por otra parte, las técnicas automáticas no son totalmente precisas, ya que dan lugar a reglas complejas y difícilmente comprensibles, anulando la propiedad de los RBSs de ejercer como interfaz entre el modo de razonamiento humano y el de procesamiento de un ordenador. Por esto las técnicas manuales y automáticas se combinan, de modo que las primeras, mediante un experto,

refinan las reglas y variables almacenadas en una base de conocimiento que ha sido obtenida con métodos automáticos. También se da el caso contrario, en el cual un experto crea la totalidad de la base de conocimiento para luego ser depurada mediante algoritmos de aprendizaje automático y minería de datos.

En ambas maneras de proceder el experto está presente durante el proceso de generación de la base de conocimiento, haciendo al RBS dependiente de él. Este hecho no supone ningún problema, más allá del coste inherente al trabajo con el experto, cuando el campo de aplicación del sistema de reglas es invariable, como lo es el control de una grúa en una cadena de montaje industrial. Este dominio de aplicación es estático, ya que la grúa y sus características no cambian con el paso del tiempo, y por ello el RBS desarrollado para su control nunca queda desactualizado. El diagnóstico médico es un dominio de aplicación dinámico, ya que el conocimiento actual sobre una enfermedad es ampliado con cada nueva investigación que se realiza sobre ella y con los casos nuevos que pueden aparecer. Un RBS que no es capaz de evolucionar a medida que lo hace su dominio de aplicación, queda obsoleto debido a que el conocimiento almacenado en él no está actualizado. Las técnicas dependientes de expertos generan bases de conocimiento estáticas, y no ofrecen ningún mecanismo para su renovación.

La generación automática de bases de conocimiento es un problema de búsqueda y optimización, donde el espacio de soluciones son todas las posibles configuraciones de bases de conocimiento, incluyendo variables y reglas, para un determinado RBS y el objetivo es encontrar la configuración más eficiente. La eficiencia de una base de conocimiento es equivalente a la del sistema basado en reglas del que forma parte y su evaluación depende del propósito del mismo. Un RBS creado para la detección de errores en una cadena de producción es más eficiente a medida que detecta más errores y genera menos falsas alarmas (falsos positivos). Al ser la generación automática de bases de conocimiento un problema que dispone de un espacio de búsqueda acotado y de criterios para evaluar cada elemento del espacio, la aplicación de los AAGG y la PG es viable.

6.2 Algoritmos Genéticos

Aplicado al proceso de construcción de un sistema basado en reglas, un AG es capaz de explorar el espacio de soluciones comprendido por la totalidad de las bases de conocimiento de un RBS de un dominio determinado. Esta técnica codifica bases de conocimiento en forma de individuos y los almacena dentro de la población de un algoritmo genético, de tal forma que, a través de los operadores de cruce, mutación y selección, se crean mejores bases de conocimiento generación tras generación hasta que alcanzar una solución óptima o válida.

Al contrario que las técnicas clásicas de generación de bases de conocimiento, tanto manuales como automáticas, los AAGG son sistemas auto-adaptativos capaces de modificar mediante evolución una base de conocimiento por alteraciones en su entorno de aplicación. Su principal desventaja radica en la codificación de bases de conocimiento como individuos de la población: un algoritmo genético utiliza individuos de longitud fija, lo cual significa que únicamente es capaz de generar bases de conocimiento con un número máximo de reglas prefijado. Al codificar una base de conocimiento, cada regla de la base de reglas se traduce en una subcadena de caracteres de longitud fija. La secuencia formada por la totalidad de subcadenas de caracteres resultantes de codificar todas las reglas de la base es la cadena que codifica la base de reglas. Si dicha cadena tiene una longitud máxima preestablecida de L subcadenas, ningún individuo generado durante la ejecución del algoritmo genético podrá superar dicha longitud, es decir, ninguna base de conocimiento podrá tener más de L reglas.

Otro inconveniente de los AAGG, derivado del sistema de codificación de individuos, es el incremento de coste computacional que supone la generación de individuos inválidos. El método de codificación de bases de reglas en cadenas de caracteres procesables por un AG da lugar a la posible generación de cadenas que codifican bases de conocimiento no válidas.

Una base de conocimiento no es válida cuando al menos una de sus reglas no está bien formada, es decir, no respeta la estructura:

IF *ANTECEDENTE* THEN *CONSECUENTE*

donde *ANTECEDENTE, CONSECUENTE* $\neq \emptyset$. Los operadores de cruce de los AAGG no cumplen la propiedad del cierre, es decir, que el cruce de individuos válidos genere a individuos igualmente válidos. Por esta razón, aunque la población inicial de un AG esté compuesta por individuos válidos, pueden aparecer individuos inválidos tras la aplicación del operador de cruce. Estos individuos no imposibilitan la ejecución del AG, ya que se les asigna un grado de adaptación (fitness) muy bajo para ser eliminados en el proceso de evolución. No obstante, el proceso de evaluación de estos individuos ralentiza la ejecución del algoritmo.

6.3 Programación Genética

Al igual que los AAGG, la PG aplicada a la generación de bases de conocimiento evoluciona una población de dichas bases para, a través de operadores de cruce, mutación y selección obtener una base de conocimiento óptima para un RBS específico. Un programa genético también es un sistema auto-adaptativo, por lo tanto no genera bases de conocimiento estáticas que pueden quedar obsoletas ante los cambios en su dominio de aplicación.

La ventaja de esta técnica evolutiva sobre los AAGG es que posibilita la creación de individuos de tamaño variable, es decir, de bases de conocimiento con un número no prefijado de reglas. Las bases de conocimiento son codificadas como individuos de la población del programa genético de manera similar a la codificación en un AG, con la salvedad de que cada individuo carece de longitud máxima. No obstante, los operadores de cruce de la PG no cumplen la propiedad del cierre, posibilitando la generación de individuos inválidos que incrementan el coste computacional. Además, la PG no impide el problema de la explosión de código.

La Programación Genética Guiada por Gramáticas es una extensión de la PG que codifica los individuos de la población como palabras de una gramática libre de contexto. En el caso de la generación de bases de conocimiento, esta gramática genera un lenguaje formado por palabras que representan bases de conocimiento existentes para un RBS determinado, codificadas para su manejo dentro de un programa genético. El operador de cruce gramatical (GBX) propio de la PGGG sí resuelve el problema del cierre y evita el problema de la explosión de código. Además, un programa genético guiado por gramáticas no puede generar individuos inválidos, ya que no son palabras pertenecientes al lenguaje generado por su gramática libre de contexto. Por tanto, la PGGG se alza como la técnica evolutiva más adecuada para el desarrollo de un sistema auto-adaptativo para la generación de bases de conocimiento basadas en reglas.

Como resultado de todo lo discutido anteriormente, se tiene por objetivo de este trabajo diseñar un modelo de generación automática de sistemas basados en reglas mediante PGGG, así como estudiar su aplicación a la detección de lesiones de rodilla a partir del análisis de curvas isocinéticas. Para ello, se ha estudiado la posibilidad de aplicar dos técnicas diferentes: la generación directa de sistemas basados en reglas de producción clásicas (RBS) y la generación indirecta de sistemas basados en reglas difusas a través de redes de neuronas difusas (FNN). Para cada uno de estos sistemas se han desarrollado los siguientes componentes: un sistema de codificación de individuos apto para la generación de su base de reglas específica, una gramática libre de contexto que permite la generación de individuos sujetos a dicha codificación y un método de evaluación de individuos especializado para el propósito del dominio de aplicación del sistema.

Un sistema de estas características se podría situar entre la primera fase del proceso de adquisición de conocimientos, que generalmente consiste en la realización de entrevistas abiertas y observación directa, y la educción de conocimiento de grano fino, realizada mediante entrevistas estructuradas y cuestionarios. De esta forma se alivia el cuello de botella que supone la educción de conocimiento del experto durante el proceso de construcción de sistemas basados en el conocimiento.

7. Generación de sistemas basados en reglas mediante PGGG

7.1 Introducción

En el presente capítulo se expone el proceso de construcción de sistemas basados en reglas cuyas bases de conocimiento son generadas automáticamente mediante PGGG, dividiéndose en dos secciones: en la primera de ellas se aborda la construcción de un sistema basado en reglas (RBS) de forma directa y en la segunda, la construcción de un sistema basado en reglas difusas (FRBS) de forma indirecta a través de una red neuronal difusa (FNN).

7.2 Generación directa de un RBS mediante PGGG

La construcción de un RBS tiene dos etapas: La generación de la base de conocimiento y la elección del sistema de inferencia junto con la estrategia de control.

7.2.1 Generación de la base de conocimiento

El proceso de AC para la construcción de una base de conocimiento es modificado por la inclusión de la PGGG, de tal forma que el sistema generador de la base de reglas mediante PGGG sustituye en parte al proceso de educción de conocimientos del experto. Como se muestra en la Figura 7.1, las técnicas tradicionales más genéricas de adquisición de conocimientos son aplicadas en primera instancia: entrevistas abiertas, observación y análisis de textos. Estas técnicas, combinadas con algoritmos de minería de datos, permiten generar una base de hechos fundamentada en la experiencia del experto y el conocimiento extraído de los datos disponibles sobre el dominio.

La base de hechos sirve de entrada al sistema automático de generación de la base de reglas mediante PGGG, durante el cual no se precisa la interacción con el experto. El contenido de la base de hechos es conformado por las variables de entrada y salida del

sistema, es decir, en su interior está definida la interfaz del RBS con el entorno. Si el propósito de un RBS es determinar el modelo de un automóvil a partir de la observación de 4 características, la base de hechos del sistema almacenará 4 variables, correspondientes a dichas características, así como una variable de salida correspondiente a la estimación del sistema sobre el modelo del automóvil.

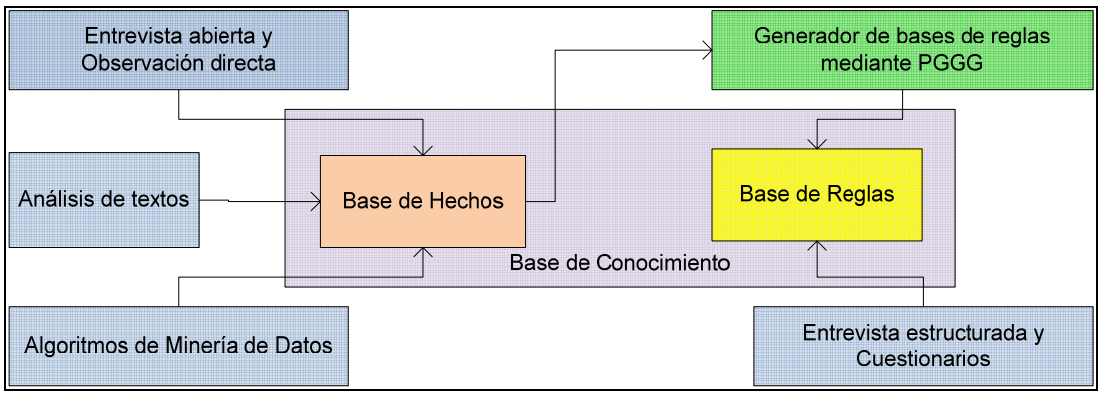


Fig. 7.1. Generación de la base de conocimento de un RBS mediante PGGG

El sistema generador de la base de reglas proporciona una base de reglas para su aplicación en el problema dado. El proceso de construcción de la base de conocimiento finaliza mediante tareas de educción de conocimiento de grano fino, como por ejemplo entrevistas estructuradas o cuestionarios al experto, con el fin de refinar la base de reglas que ha sido generada automáticamente.

7.2.1.1 Sistema generador de bases de reglas para un RBS mediante PGGG

Un programa genético parte de una población inicial de individuos generados aleatoriamente para, mediante la aplicación de los operadores de cruce, mutación, selección y reemplazo a lo largo de un número finito de generaciones, encontrar el individuo óptimo para la resolución de un problema. Aplicado a la generación de bases de reglas, cada individuo de la población codifica una base de reglas completa aplicable a un RBS específico.

- **Gramática libre de contexto:**

  Todos los individuos creados, bien aleatoriamente o bien como resultado de un cruce o mutación, durante la ejecución del programa genético son árboles de derivación generados por una gramática libre de contexto, desarrollada específicamente para el RBS. Dicha gramática libre de contexto G se define mediante un vector de cuatro elementos que sigue la siguiente estructura:

$$G = (\Sigma_N, \Sigma_T, S, P)$$

  tal que $\Sigma_N \cap \Sigma_T = \emptyset$, donde Σ_N es el alfabeto de símbolos no terminales, Σ_T es el alfabeto de símbolos terminales, S el axioma y P el conjunto de producciones escritas en la forma Backus-Naur.

  Una gramática libre de contexto adaptada al ejemplo del RBS diseñado para la clasificación de automóviles según el modelo, debe generar el lenguaje de las bases que contienen reglas del tipo:

IF *ANTECEDENTE* THEN *CONSECUENTE*

  donde *ANTECEDENTE* es un conjunto de asignaciones de valores a la variables de entrada al sistema y *CONSECUENTE*, la asignación de un valor a la variable de salida del sistema, todas ellas contenidas en la base de hechos y descritas en el marco conceptual. El antecedente puede contener más de una asignación, estando en tal caso unidas mediante los operadores lógicos de conjunción (AND) y de disyunción (OR) indistintamente. Las cláusulas del antecedente pueden aparecer en forma negada mediante el operador NOT.

$G1 = (\Sigma_N, \Sigma_T, S, P)$

$\sum_N$ = { S, I, REGLA, ANTECEDENTE, CONSECUENTE, EXPR, EXP1, EXP2, EXP3, EXP4, OPR1, OPR2, MODELO, COMBUSTIBLE }

$\sum_T$ = { if, then, not, and, or, modelo, cilindrada, combustible, peso, valor, diésel, gasolina, real, modelo1, modelo2, modelo3, modelo4, =, <, >, ≤, ≥, (,), ; }

P = { S::= I,
I ::= REGLA I | REGLA,
REGLA ::= if (ANTECEDENTE) then CONSECUENTE,
CONSECUENTE ::= (modelo = MODELO),
ANTECEDENTE ::= EXPR | ANTECEDENTE OPR1 EXPR
EXPR ::= EXP1 | EXP2 | EXP3 | EXP4,
EXP1 ::= (cilindrada OPR2 real) | (not (cilindrada OPR2 real))
EXP2 ::= (combustible = COMBUSTIBLE) |
(not (combustible = COMBUSTIBLE)),
EXP3 ::= (peso OPR2 real) | (not (peso OPR2 real)),
EXP4 ::= (valor OPR2 real) | (not (valor OPR2 real)),
OPR1 ::= and | or,
OPR2 ::= < | > | ≤ | ≥ | =,
COMBUSTIBLE ::= diésel | gasolina,
MODELO ::= modelo1 | modelo2 | modelo3 | modelo4 }

Fig. 7.2. Gramática libre de contexto G1 para un RBS clasificador de automóviles

La Figura 7.2 muestra la gramática libre de contexto G1 para el RBS de clasificación de automóviles, siendo “cilindrada”, “combustible”, “peso” y “valor” las variables de entrada al sistema y “modelo” la variable de salida. Todas las variables de entrada toman valores dentro del conjunto de los números reales,

excepto “combustible”, que únicamente puede tomar los valores “diesel” y “gasolina”. La variable de salida puede tomar los valores “modelo1”, “modelo2”, “modelo3” y “modelo4”, relativos a los cuatro modelos de automóvil contemplados por el RBS. La Figura 7.3 contiene una base de cuatro reglas generada a partir de la gramática G1.

R1	if ((cilindrada < real) and (combustible = diésel)) then (modelo = modelo1)
R2	if ((not (valor > real))) then (modelo = modelo2)
R3	if ((peso > real) and (peso < real)) then (modelo = modelo3)
R4	if ((cilindrada > real) or (valor ≥ real)) then (modelo = modelo4)

Fig. 7.3. Base de reglas generada a partir de G1

Toda base de conocimiento generada por G1 es una palabra perteneciente a su lenguaje, y todas las palabras del lenguaje se obtienen a partir de un árbol de derivación. El árbol de derivación de la base de reglas de la Figura 7.3 está representado en la Figura 7.4.

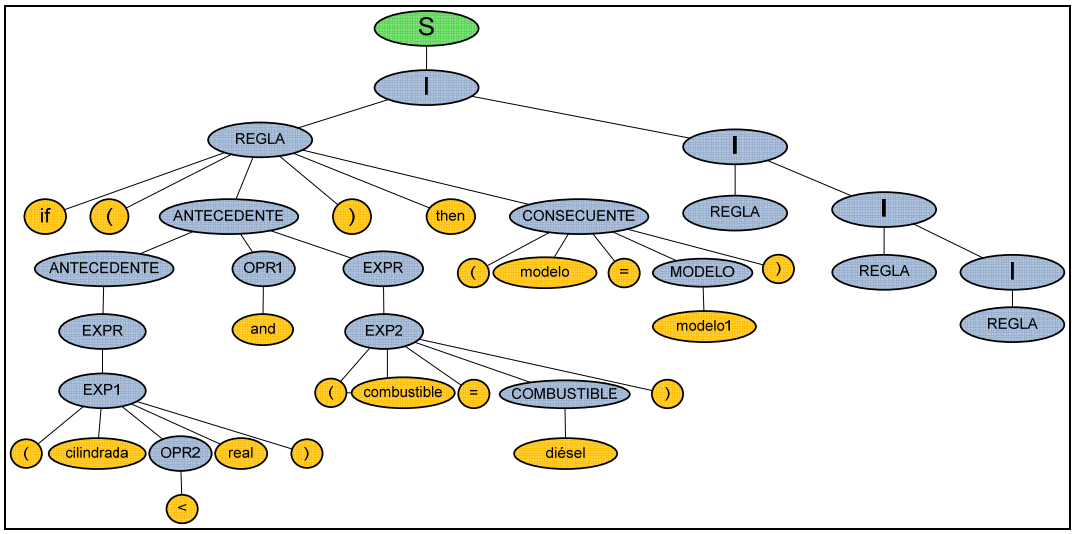


Fig. 7.4. Árbol de derivación para la base de reglas de la Figura 7.3

El nodo raíz, de color verde, corresponde al axioma de la gramática; los nodos azules son símbolos no terminales y los nodos amarillos símbolos terminales. Para no sobrecargar la figura, sólo se muestra desplegado el nodo no terminal correspondiente a la regla R1. Los nodos correspondientes a las tres reglas restantes se desarrollan de manera análoga, encadenando producciones hasta alcanzar los nodos terminales que componen las reglas R2, R3 y R4. La base de conocimiento se extrae recorriendo el árbol de derivación en profundidad y seleccionando los nodos terminales de izquierda a derecha.

- **Gradiente evolutivo:**

  Las variables definidas en la base de hechos que toman valores pertenecientes a un conjunto son definidas en la gramática como parte de los símbolos terminales de la misma, por ejemplo: la variable de salida "modelo" genera cinco símbolos terminales correspondientes al nombre de la variable y a cada uno de sus cuatro posibles valores: "modelo1", "modelo2", "modelo3" y "modelo4". Las variables que toman valores dentro de un conjunto no finito, como los números reales, no pueden ser definidas de esa forma, ya que no se puede generar un símbolo terminal para cada número real. Por esta razón se define dentro de la gramática un símbolo terminal llamado "real", que indica al sistema generador de la base de reglas que toda variable asociada él toma valores dentro del conjunto de los números reales. En el ejemplo dado, las variables "cilindrada", "peso" y "valor" son asociadas con el símbolo terminal "real".

  El valor de cada variable asociada al terminal "real" es calculado por el sistema al comenzar el programa genético mediante la media aritmética muestral. Dicho valor evoluciona paralelamente a la población del programa genético a través del método del gradiente evolutivo. Este método introduce una mejora al operador de cruce gramatical GBX usado por el sistema generador de la base de reglas de la siguiente forma:

1. Siendo P1 y P2 dos individuos seleccionados para el cruce, se aplica el algoritmo GBX.
2. Sea O1 el primer descendiente del cruce. Se recorre su árbol de derivación buscando los nodos no terminales EXPR que derivan en una expresión que alberga el símbolo terminal "real". Sea el conjunto C compuesto por las cláusulas de los antecedentes de las reglas que albergan el símbolo terminal "real".
3. Para cada cláusula $C_i \in C$, se calcula un valor aleatorio que, si supera un umbral prefijado, indica la aplicación del gradiente evolutivo.
4. Si se aplica el gradiente evolutivo, se calcula aleatoriamente el valor $r \in (0, 1]$ y el valor $sig \in [+, -]$. En caso contrario, se continúa en 7.
5. Se calcula el valor $real_i$, que equivale al valor asociado al terminal "real" de la cláusula C_i.
6. Se calcula el valor $real_i' = real_i + (sig)\ r$, y se asigna al terminal real de la cláusula C_i.
7. Si quedan cláusulas en C, se continúa en 3. Si no, se continúa en 2 con el segundo descendiente del cruce: O2.

La figura 7.5 muestra un ejemplo de la aplicación del gradiente evolutivo sobre una cláusula perteneciente al antecedente de un descendiente. El valor original asociado al terminal real es 1500, modificándose a 1499.1 tras la aplicación del gradiente evolutivo. El gradiente evolutivo refina los valores calculados inicialmente para todas las variables asociadas a un símbolo terminal "real". Su efecto contribuye a la evolución de la población de individuos, ya que aquellos cuya adaptación ha empeorado a causa de la aplicación del gradiente son proclives a ser eliminados de la población. Por el contrario, los individuos que permanecen son aquellos que no han sido alterados por el gradiente o que han mejorado su adaptación tras la modificación.

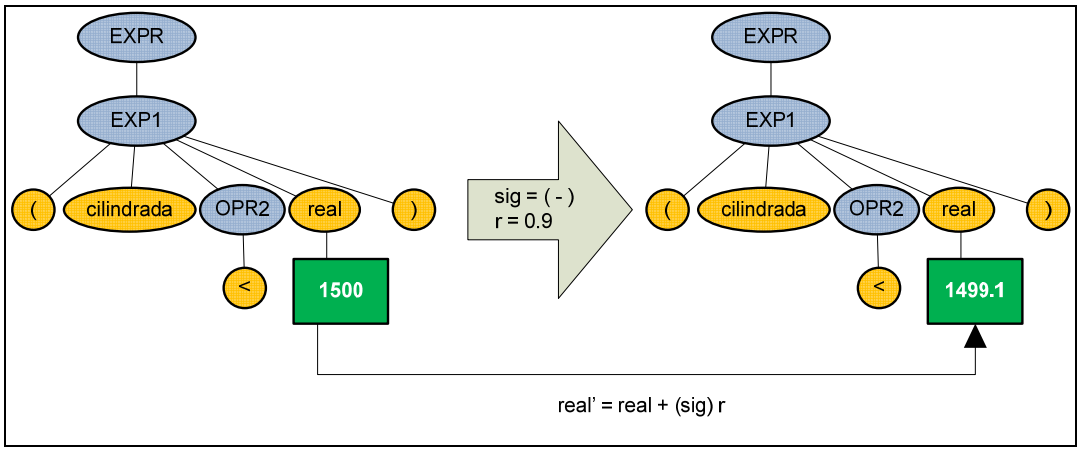


Fig. 7.5. Ejemplo de aplicación del gradiente evolutivo.

- **Evaluación de los individuos:**

  El cálculo de la adaptación en el sistema generador de la base de reglas consiste en la descodificación de cada individuo de la población en una base de reglas, su inserción en la base de conocimiento del RBS y la evaluación de su comportamiento frente a un conjunto de datos de entrenamiento. Los datos o patrones de entrenamiento son un conjunto de instancias de las variables de entrada al RBS, cada una de las cuales tiene asociado un valor de la variable de salida del sistema. En el ejemplo de la clasificación de automóviles, un conjunto de entrenamiento tiene la siguiente estructura:

  Patrón1 = { cilindrada = 1500, combustible = diésel, peso = 2000, valor = 26000; modelo = modelo1 }

  Patrón2 = { cilindrada = 3200, combustible = gasolina, peso = 2200, valor = 42500; modelo = modelo2 }

  Patrón3 = { cilindrada = 2000, combustible = diésel, peso = 3000, valor = 25000; modelo = modelo3 }

  Patrón4 = { cilindrada = 2500, combustible = diésel, peso = 4500, valor = 36000; modelo = modelo4 }

El RBS clasifica, conociendo únicamente el valor de las variables de entrada, cada patrón de entrenamiento haciendo uso de la base de reglas cuya adaptación se está evaluando.

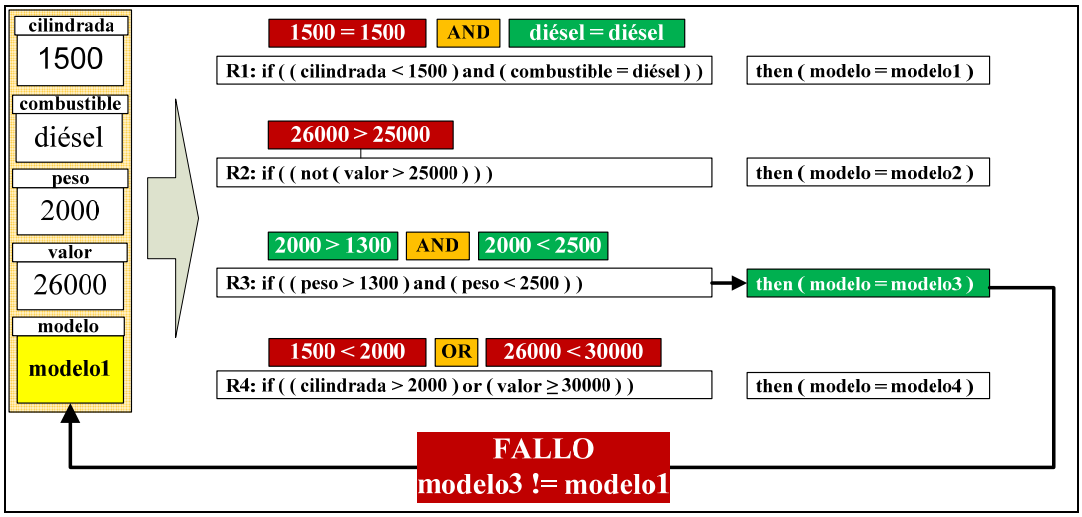


Fig. 7.6. Proceso de inferencia en el RBS clasificador de autmóviles.

Como se muestra en la Figura 7.6, el resultado de esa clasificación es comparado con el valor de la variable de salida del patrón de entrenamiento, anotando un acierto cada vez que coincida y un fallo cuando difiera. Existe la posibilidad de que el RBS no emita ninguna clasificación para un patrón de entrenamiento, debido a que los valores de las variables de entrada recogidos en dicho patrón no han activado ninguna regla. La adaptación de la base de reglas evaluada será mejor cuantos más automóviles clasifique correctamente.

7.2.2 Sistema de inferencia y estrategia de control

Los criterios de selección del sistema de inferencia y de la estrategia de control del RBS no varían con la incorporación del sistema generador de reglas mediante programación genética. Su elección sigue siendo directamente dependiente del dominio de aplicación del RBS así como de los costes asimilables en su proceso de desarrollo. Para el clasificador de automóviles, el sistema de inferencia propicio debe usar encadenamiento hacia delante: las reglas generadas automáticamente para el RBS dotan al sistema de

amplitud de entrada, ya que relacionan cuatro conclusiones (modelo1, modelo2, modelo3 y modelo4) con múltiples combinaciones de hechos de entrada al sistema. Este RBS no necesita de una estrategia de control que defina el orden de activación de sus reglas, ya que dicho orden no es relevante para clasificar un patrón o conjunto de hechos de entrada. Sin embargo, todas las reglas activadas ante un mismo patrón generan una salida, lo que deviene en un conjunto de salidas cuyo tamaño es el número de reglas activadas. Dicho conjunto de salidas debe ser traducido en una de las cuatro conclusiones de salida del sistema, correspondientes a los cuatro modelos de automóvil contemplados. La estrategia de control debe encargarse de definir el método de agregación de las salidas generadas por el motor de inferencia de tal forma que la salida del RBS sea única.

7.3 Generación de un FRBS a través de una FNN mediante PGGG

El proceso de construcción de una FNN que representa la base de reglas difusas de un FRBS se divide en la generación de la base de conocimiento y del motor de inferencia, no existiendo una estrategia de control.

7.3.1 Generación de la base de conocimiento

En primer lugar, es necesario definir el marco conceptual y la base de hechos a partir de entrevistas abiertas y observación directa del experto, análisis de textos y algoritmos de minería de datos. De este proceso de educción se pueden obtener las variables lingüísticas de entrada y salida al sistema y, opcionalmente, los conjuntos difusos con sus etiquetas lingüísticas asociadas y las funciones de pertenencia que los definen. Cuando esto último no es posible se pueden emplear algoritmos de agrupación o clustering, los cuales establecen asociaciones dentro de los datos disponibles que dan lugar a los conjuntos difusos. Uno de los algoritmos de clustering más utilizados es el Fuzzy C-Means (FCM). Este algoritmo agrupa una distribución de datos en un número prefijado C de conjuntos o clusters difusos, los cuales tienen forma de hiper-esferas. Cada dato de la distribución pertenece a cada uno de dichos conjuntos en un cierto

grado, llamado valor de pertenencia, el cual depende de la distancia de dicho dato al centro del conjunto. El objetivo del algoritmo es encontrar los valores de los centros de los C conjuntos que minimizan los valores de pertenencia de los datos. El FCM finaliza cuando el mayor valor de pertenencia calculado es menor que un umbral predefinido ε, generando como salida los centros de los conjuntos difusos.

Abordar el problema de la clasificación de automóviles en modelos mediante un sistema basado en reglas difusas codificado en una FNN, requiere establecer conjuntos difusos para cada una de las variables de entrada y salida: cilindrada, combustible, peso, valor y modelo. Las variables que toman valor dentro de un conjunto no finito, como peso, son agrupadas mediante clustering en un número predefinido de conjuntos o clusters. Por ejemplo, la variable "peso" toma valores reales en el intervalo [0, 3000] y está relacionada con los conjuntos difusos "ligero" y "pesado". Sus funciones de pertenencia están reflejadas en la Figura 7.7.

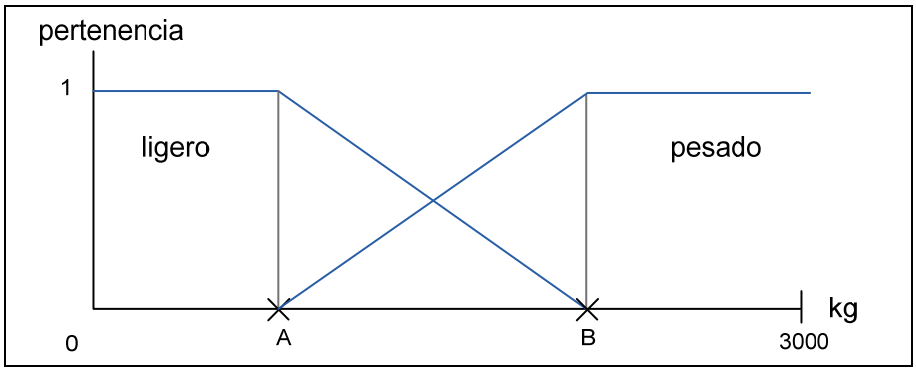


Fig. 7.7. Funciones de pertenencia a los conjuntos difusos "ligero" y "pesado".

La variable "combustible" no es difusa, ya que sus posibles valores "diésel" y "gasolina" no son borrosos. Esta variable no tiene funciones de pertenencia y no realiza ningún proceso de borrosificación para los valores de entrada al sistema. Para que la FNN pueda operar con este tipo de variables, se debe realizar una transformación de tal forma que la variable tome valores reales dentro de un intervalo acotado. Sobre ese intervalo se establecen dos funciones de pertenencia a dos conjuntos difusos llamados "diésel" y "gasolina" que no dejan margen de incertidumbre.

La Figura 7.8 muestra un ejemplo para la representación de este intervalo. La transformación de la variable "combustible" se extiende a los datos del conjunto de entrenamiento, en donde los patrones cuya variable "combustible" toma el valor "diésel", ahora debe tomar el valor "A", y aquellos con valor "gasolina" ahora tienen valor "C", siendo A,B,C ∈ [0,1].

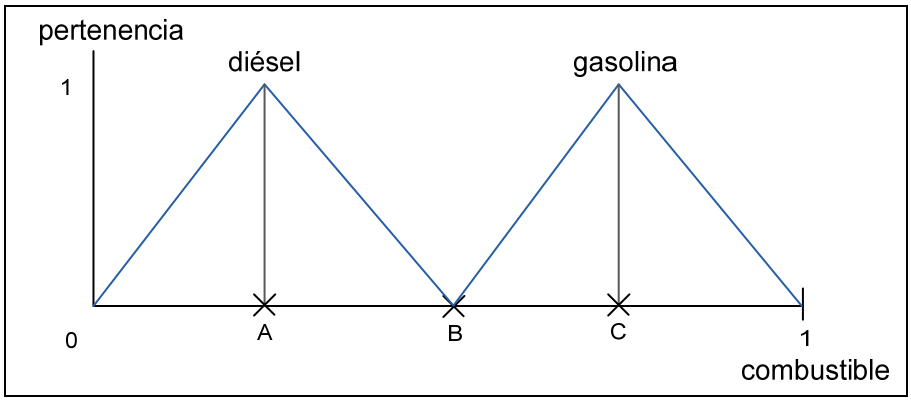


Fig. 7.8. Funciones de pertenencia a los conjuntos difusos "diésel" y "gasolina".

Esta base de hechos sirve de entrada a un sistema que, mediante PGGG, crea una FNN que representa la base de reglas difusas de un FRBS. Para finalizar el proceso se realizan de entrevistas estructuradas y cuestionarios con el experto, cuyo fin es refinar dicha base de reglas obtenida automáticamente.

7.3.1.1 Sistema generador de bases de reglas difusas codificadas en FNNs mediante PGGG

Las diferencias que presenta este sistema con respecto al generador de bases de reglas para RBSs son: la codificación de los individuos de la población, la gramática libre de contexto y en el método de evaluación de la adaptación de los individuos.

- **Codificación de los individuos:**

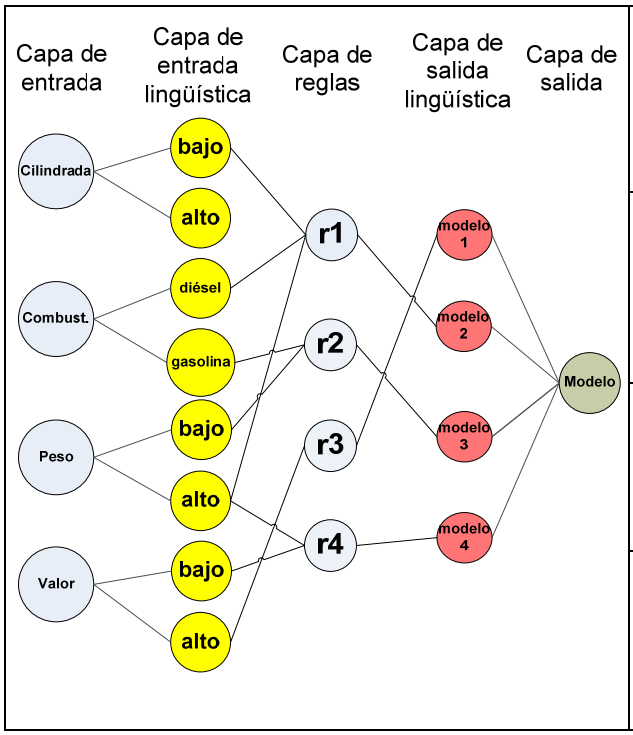
  La gramática debe generar un lenguaje formado por palabras que codifican topologías neuronales que representan bases de reglas difusas. La Figura 7.9 muestra una base de cuatro reglas codificada en una FNN para su aplicación al problema de clasificación de automóviles, teniendo en cuenta los siguientes aspectos:

  1. Cada variable real, tanto de entrada como de salida del sistema, se borrosifica en una variable lingüística que lleva su mismo nombre.

  2. Cada variable lingüística de entrada está asociada a dos conjuntos difusos, cuyas etiquetas lingüísticas son:
     - Cilindrada: [bajo, alto]
     - Combustible: [diésel, gasolina]
     - Peso: [bajo, alto]
     - Valor: [bajo, alto]

     El tamaño de la capa de entrada es cuatro (I = 4). La capa de entrada lingüística alberga dos neuronas por cada neurona de la capa de entrada (C = 2), por tanto su tamaño es ocho (I×C).

  3. La variable de salida lingüística "modelo" está asociada a cuatro conjuntos difusos (K = 4) con las siguientes etiquetas lingüísticas: [modelo1, modelo2, modelo3, modelo4]. La capa de salida únicamente consta de una neurona (O = 1), y la capa de salida lingüística de cuatro (O×K).



r1	IF Cilindrada IS bajo AND Combustible IS diésel AND Peso IS alto THEN Modelo IS modelo2
r2	IF Combustible IS gasolina AND Peso IS bajo THEN Modelo IS modelo3
r3	IF Valor IS alto THEN Modelo IS modelo1
r4	IF Peso IS alto AND Valor IS bajo THEN Modelo IS modelo4

Fig. 7.9. FNN clasificadora de automóviles.

Codificar FNNs que representan bases de reglas difusas directamente como árboles de derivación de un lenguaje da lugar a gramáticas muy grandes y complejas, que deben incluir símbolos terminales para todos los conjuntos difusos además de para cada variable lingüística. Para paliar este inconveniente se realiza una codificación intermedia, en la que cada base de reglas es traducida a una cadena binaria compuesta por R segmentos de un número fijo de bits, donde R es el número de reglas de la base y cada bit representa la presencia (1) o ausencia (0) de una cláusula.

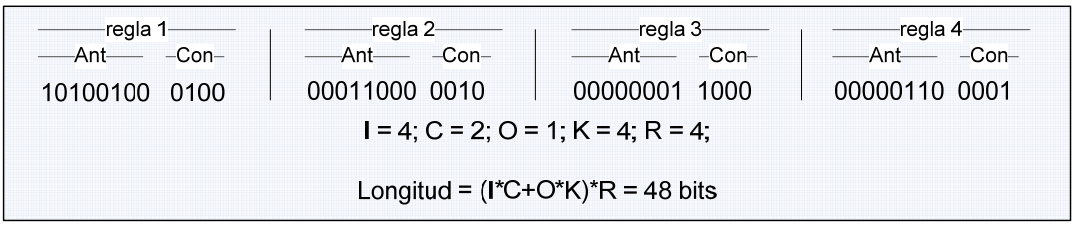


Fig. 7.10. Base de reglas difusas de la Figura 7.9 codificada en una cadena binaria.

Cada regla se codifica en una subcadena de tamaño fijo igual a la suma del número de neuronas en la capa de entrada lingüística y de la capa de salida lingüística (I×C+O×K). La longitud total de la cadena que codifica la base de reglas es el resultado de la expresión (I×C+O×K) × R. La Figura 7.10 muestra la cadena binaria resultante de codificar la base de reglas de la Figura 7.9.

- **Gramática libre de contexto:**

  La gramática libre de contexto G2 que genera el lenguaje de bases de reglas para un FRBS representadas mediante FNNs, aplicada al problema de clasificación de automóviles, se describe en la Figura 7.10.

$G2 = (\Sigma_N, \Sigma_T, S, P)$

$\sum_N = \{ S, R, H, A, B, C, Z \}$

$\sum_T = \{ 0, 1\}$

P = { S ::= I,
I ::= I H | H,
H ::= A Z,
A ::= C B B B | B C B B | B B C B | B B B C,
B ::= 0 0 | 0 1 | 1 0,
C ::= 0 1 | 1 0,
Z ::= 1 0 0 0 | 0 1 0 0 | 0 0 1 0 | 0 0 0 1 }

Fig. 7.11. Gramática libre de contexto que genera bases de reglas para la FNN clasificadora de automóviles: I = 4, C = 2, O = 1 y K = 4.

El árbol de derivación correspondiente a la cadena binaria de la Figura 7.10 está representado en la Figura 7.12. Para reducir la complejidad de la figura, únicamente se despliega el nodo no terminal correspondiente a la primera regla de la base de reglas, siendo análogo el desarrollo de los nodos correspondientes a las tres reglas restantes.

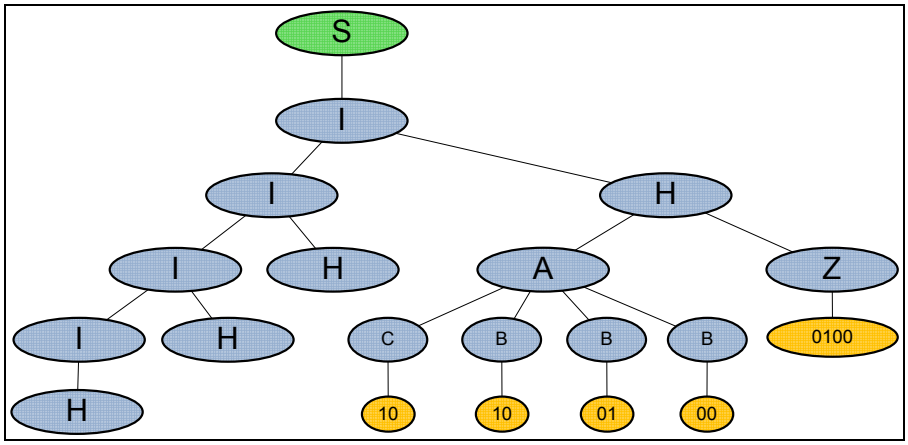


Fig. 7.12. Árbol de derivación para la base de reglas codificada en el individuo de la Figura 7.10.

- **Evaluación de los individuos:**

  La base de reglas representada mediante una FNN es introducida en el FRBS, donde se analiza su comportamiento al clasificar un conjunto de datos de entrenamiento. Para cada patrón del conjunto de entrenamiento se sigue el proceso descrito a continuación:

  1. Sea r1 la primera regla de la base de reglas de la FNN y p1 el patrón de entrenamiento.
  2. Se asigna a cada neurona de entrada de la FNN su valor correspondiente almacenado en p1.
  3. Se calcula la pertenencia de los valores de las neuronas de entrada a sus conjuntos difusos correspondientes, siempre y cuando aparezcan en el antecedente de r1.
  4. Se realiza aplica el operador de conjunción sobre los valores obtenidos en el paso anterior, obteniendo el valor de activación de r1 (h_1).

5. Se calcula, mediante la regla composicional de inferencia y el operador de implicación, la forma del conjunto difuso indicado en el consecuente de r1.
6. Si hay existen más reglas en la base de reglas, se continúa con la siguiente en 2. Si no, se continúa en 7.
7. Se aplica la desborrosificación sobre los conjuntos difusos proporcionados por cada regla. El resultado es la salida real del sistema (y).

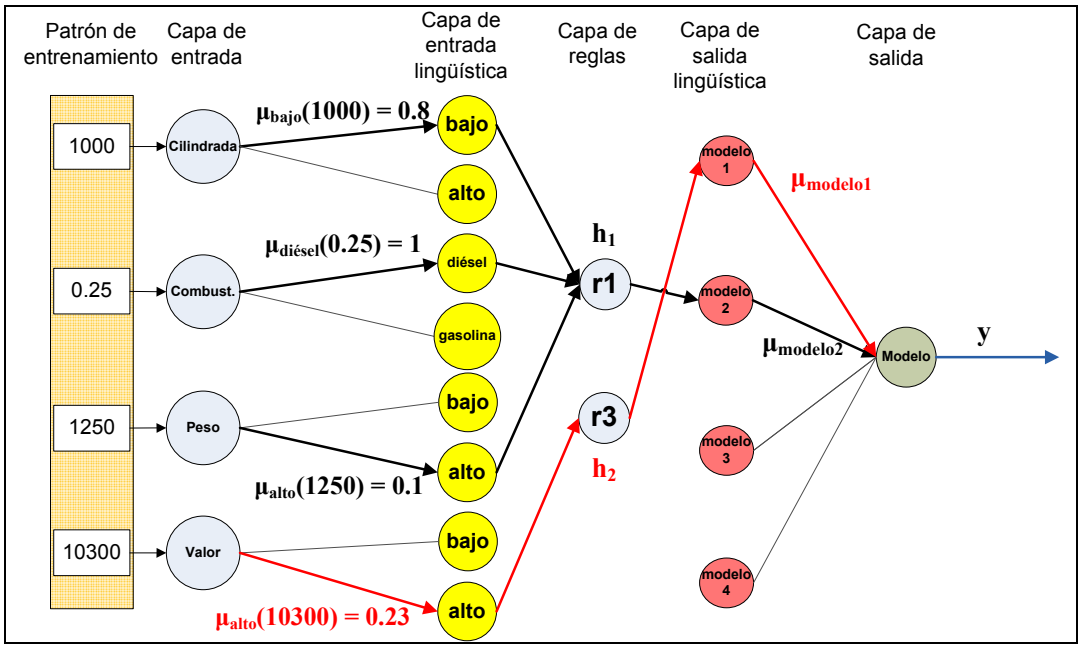


Fig. 7.13 Inferencia en la FNN clasificadora de automóviles.

La Figura 7.13 ejemplifica el proceso de inferencia de la FNN aplicada a la clasificación de automóviles, considerando únicamente la existencia de las reglas r1 y r3. La salida deseada de la red para el problema no es un valor real, sino un valor lingüístico correspondiente al modelo del coche. Dicho valor se obtiene calculando la pertenencia del valor real a los cuatro conjuntos difusos de salida del sistema ($\mu_{modelo1}$, $\mu_{modelo2}$, $\mu_{modelo3}$, $\mu_{modelo4}$) y seleccionado la etiqueta lingüística del conjunto para el que haya obtenido mayor pertenencia. Si el valor lingüístico de salida coincide con el del patrón de entrenamiento clasificado, se anota un acierto y, en otro caso, se anota un fallo. Un número alto de aciertos indica un grado de adaptación alto de la base de reglas.

7.3.2 Diseño del motor de inferencia

El diseño del motor de inferencia supone la elección del modo de desborrosificación y los operadores de implicación, conjunción y agregación. La principal diferencia entre los métodos de desborrosificación, FATI y FITA, es que el segundo de ellos es computacionalmente menos costoso que el primero. No obstante, la potencia proporcionada por los ordenadores actuales hace que este hecho sea despreciable. Respecto a los operadores, no existe ningún factor relevante que guíe en su elección. Sin embargo, dependiendo de los operadores y el modo de borrosificación seleccionados, la salida del sistema para un mismo patrón de entrenamiento varía. Por ello, es recomendable realizar varias pruebas sobre un mismo conjunto de entrenamiento con diferentes combinaciones, con el fin de averiguar cuál aporta mejores resultados.

8. Resultados

En este capítulo se presentan los resultados de la aplicación de los dos sistemas generadores de reglas descritos en el capítulo anterior a un problema de detección de lesiones de rodilla a través de curvas isocinéticas.

8.1 Descripción del problema

Una máquina isocinética registra la fuerza muscular realizada por un paciente durante la realización de un ejercicio, produciendo una serie de valores distribuidos a lo largo de la duración de dicho ejercicio. Esta información resulta útil para su aplicación en múltiples dominios, tales como el diagnóstico muscular, la rehabilitación, la prevención de lesiones, la planificación y evaluación de entrenamientos, etc… La tecnología existente únicamente provee funcionalidad para representar gráficamente la información recogida por la máquina, de tal forma que los expertos en el dominio son los encargados de su análisis e interpretación. Este proceso es ineficiente ya que el experto, generalmente un fisioterapeuta o un médico especialista, debe basarse en su propia experiencia y conocimiento para tomar decisiones en base a la información isocinética, debido a la escasez de modelos de referencia existentes para las lesiones más comunes.

El presente problema se centra en hacer uso de la información recogida por una máquina isocinética de rodilla para elaborar un sistema de detección de lesiones en dicha articulación. Los ejercicios realizables sobre esta máquina consisten en movimientos de rodilla de velocidad fija predeterminada expresada en grados por segundo. Durante la ejecución de un ejercicio, la máquina aplica la resistencia necesaria para asegurar que se mantiene la velocidad prefijada y, finalizado el ejercicio, provee la distribución de la fuerza aplicada por el paciente a lo largo del tiempo de ejecución. Así mismo, se incluye información adicional sobre el ángulo de flexión de la rodilla (Figura 8.1).

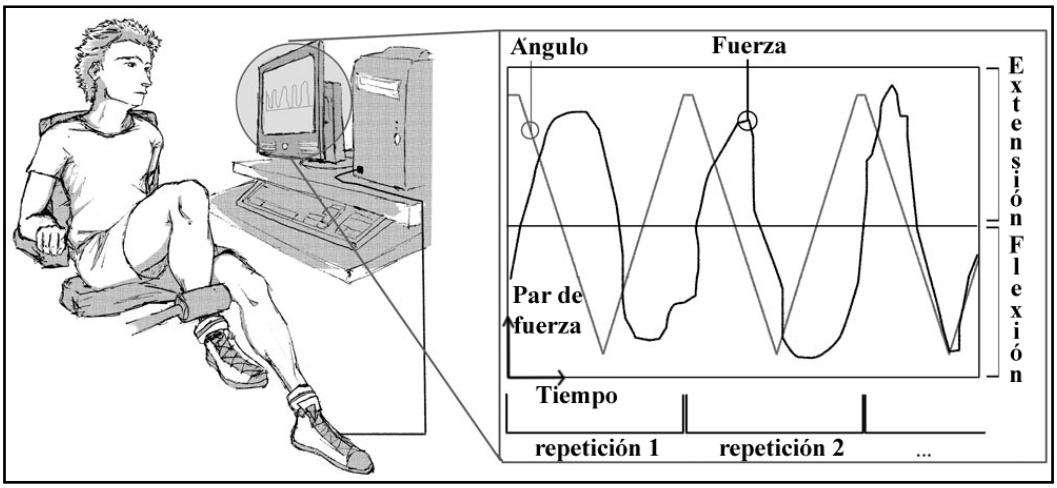


Fig. 8.1. Funcionamiento de una máquina isocinética para la rodilla.

Las dos series temporales resultantes de un ejercicio se denominan curva isocinética. Cada curva isocinética se compone de un número indeterminado de repeticiones de un movimiento de rodilla compuesto por dos fases: extensión y flexión.

8.2 Generación automática de sistemas detectores de lesiones de rodilla mediante PGGG

Aplicada a este dominio, la generación automática de sistemas basados en reglas mediante PGGG permite la construcción de un sistema capaz de detectar lesiones de rodilla en pacientes que han realizado ejercicios isocinéticos, analizando las curvas proporcionadas por la máquina isocinética. Para que una curva isocinética de tamaño variable sea analizable por dicho sistema, ha de ser procesada y transformada en un vector de características de dimensión fija. Como resultado de este proceso se han obtenido las características que se listan en la Figura 8.2. El conocimiento necesario para abordar esta tarea ha sido adquirido mediante análisis de textos especializados y entrevistas con un experto en el dominio, perteneciente al Centro de Medicina Deportiva del Ministerio de Educación, Política Social y Deporte.

Nº	Nombre	Descripción
1	secDifParMax	La secuencia resultante de la diferencia entre los pares máximos que se producen en el ejercicio, codificada en un único número mediante el teorema fundamental de la aritmética (TFA).
2	secDifAngParMax	La secuencia resultante de la diferencia entre los ángulos en que se producen los pares máximos, codificada en un único número mediante TFA.
3	secDifParMin	La secuencia resultante de la diferencia entre los pares mínimos que se producen en el ejercicio, codificada en un único número mediante TFA.
4	secDifAngParMin	La secuencia resultante de la diferencia entre los ángulos en que se producen los pares mínimos, codificada en un único número mediante TFA.
5	parMax	Par máximo del ejercicio
6	angParMax	Angulo del par máximo del ejercicio
7	timParMax	Tiempo en el que se produce el par máximo
8	parMin	Par mínimo del ejercicio
9	angParMin	Angulo del par mínimo del ejercicio
10	timParMin	Tiempo en el que se produce el par mínimo
11	timMedParMaxExt	Tiempo medio en llegar al máximo en las extensiones de un ejercicio
12	timMedParMinFlx	Tiempo medio en llegar al mínimo en las flexiones de un ejercicio
13	parMedExt	Par medio en las extensiones de un ejercicio
14	parMedFlx	Par medio en las flexiones de un ejercicio
15	desTipTimMaxExt	Desviación típica muestral del tiempo medio en llegar al máximo en las extensiones de un ejercicio
16	desTipTimMinFlx	Desviación típica muestral del tiempo medio en llegar al mínimo en las flexiones de un ejercicio
17	desTipParExt	Desviación típica muestral del par medio de las extensiones en un ejercicio
18	desTipParFlx	Desviación típica muestral del par medio de las flexiones en un ejercicio

Fig. 8.2. Características extraídas del procesamiento de una curva isocinética.

El teorema fundamental de la aritmética o teorema de factorización única, usado para extraer las cuatro primeras características, es ejemplificado en la Figura 8.3 mediante el cálculo de la característica "secDifParMax". En primer lugar se obtienen las diferencias de los pares de fuerza máximos en repeticiones adyacentes. Cada una de esas diferencias es usada en orden de adquisición como exponente de un número primo, comenzando por el número 2. Las potencias de los números primos se multiplican para obtener el valor de la característica "secDifParMax".

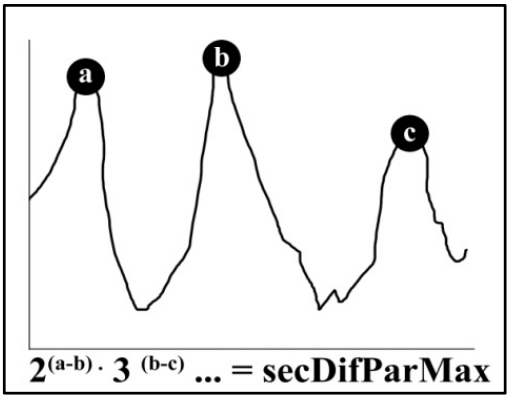


Fig. 8.3. Extracción de la característica "secDifParMax" mediante TFA.

Al conjunto de curvas isocinéticas usadas para construir el sistema basado en reglas para la detección de lesiones se denomina conjunto de entrenamiento. Tras el análisis y procesado de dicho conjunto de curvas, se obtiene un conjunto de idéntico tamaño de vectores de características, que conforma la entrada al generador automático del sistema basado en reglas mediante PGGG. Esta fase de desarrollo es llamada la fase de entrenamiento, y provee como resultado la salida del programa genético: un sistema basado en reglas capaz de detectar lesiones de rodilla a partir de curvas isocinéticas. Dicho sistema es sometido a una fase de pruebas, en la cual debe probar su eficacia detectando lesiones en curvas isocinéticas, previamente transformadas en vectores de características, que no le han sido presentadas durante la fase de entrenamiento.

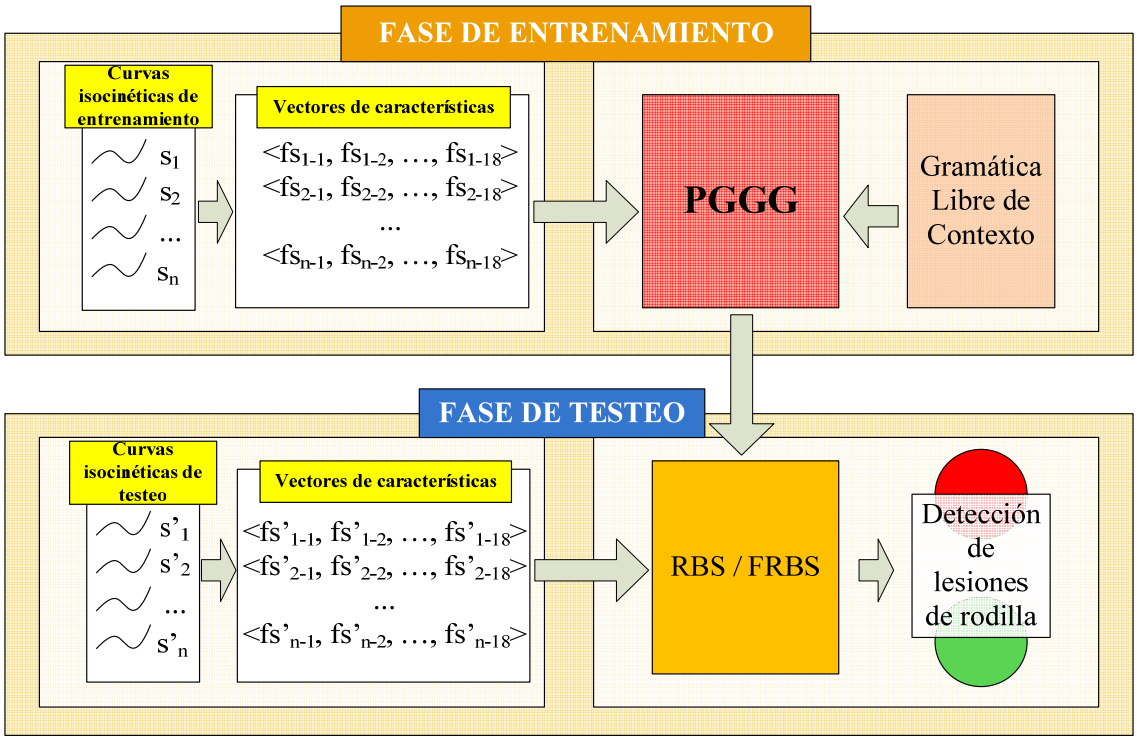


Fig. 8.3. Generación automática de un sistema basado en reglas para la detección de lesiones de rodilla mediante PGGG.

Los procesos de creación de las dos técnicas propuestas en este trabajo, RBS y FRBS, esquematizados en la Figura 8.3, tienen los siguientes aspectos comunes en la configuración del PGGG:

- Algoritmo GPBG para la generación de la población inicial con profundidad máxima 15.
- El tamaño de la población es 20.
- El número máximo de generaciones es 2000.
- Operador de cruce GBX.
- Operador de selección por torneo.
- La adaptación de un individuo de la población es igual al número de aciertos del sistema que hace uso de la base de reglas codificada en dicho individuo, dividido entre el número de patrones evaluados. Su valor varía en el intervalo [0, 1].
- Reemplazo de individuos elitista: únicamente los cuatro mejores individuos de una generación pasan a la siguiente.
- No se aplica mutación.

El conjunto de datos proporcionado por el experto del dominio asciende a un total de 92 curvas isocinéticas, divididas en un conjunto de 72 curvas para la fase de entrenamiento y 20 curvas para la fase de pruebas. A continuación, se describen los aspectos de configuración y las características de cada técnica, RBS y FRBS, acompañados de los resultados de cada una de ellas para el conjunto de datos mencionado.

8.3 Generación automática de un RBS para detectar lesiones de rodilla mediante PGGG

8.3.1 Configuración específica

La Figura 8.4 muestra la gramática G_{RBS} usada por el generador automático de RBSs aplicado al problema de detección de lesiones de rodilla. El alfabeto de símbolos terminales contiene las partículas y operadores necesarios para la construcción de las reglas condicionales, las características que componen los vectores extraídos de las curvas isocinéticas, la variable "pronóstico", que es la salida del sistema, y sus posibles valores: "lesión" y "normal".

Las reglas de un RBS aplicado a la detección de lesiones de rodilla relacionan dos posibles salidas (lesión, normal) con múltiples combinaciones de hechos de entrada, por lo que dotan al sistema de amplitud de entrada. Por esto, el modo de encadenamiento apropiado para realizar inferencia es hacia delante. La estrategia de control, que agrega las salidas de las diferentes reglas activadas para un mismo patrón de entrada, es la siguiente: se cuenta el número de reglas que clasifican a dicho patrón como normal y como lesión. La cantidad más alta define la salida del sistema para el patrón. En caso de empate o de no haberse activado ninguna regla, el sistema no emite clasificación.

$G_{RBS} = (\Sigma_N, \Sigma_T, S, P)$

$\sum_N$ = { S, REGLA, ANTECEDENTE, CONSECUENTE, EXP, E, OPR, OPR2, CLÁUSULA, PRONÓSTICO }

$\sum_T$ = { if, then, not, and, or, secDifParMax, secDifAngParMax, secDifParMin, secDifAngParMin, parMax, angParMax, timParMax, parMin, angParMin, timParMin, timMedParMaxExt, desTipTimMaxExt, timMedParMinFlx, desTipTimMinFlx, parMedExt, desTipParExt, parMedFlx, desTipParFlx, real, pronóstico, normal, lesión, =, <, >, ≤, ≥, (,), ; }

P = { S::= REGLA,
REGLA ::= REGLA REGLA,
REGLA ::= if ANTECEDENTE then CONSECUENTE,
CONSECUENTE ::= (pronóstico = PRONÓSTICO); ,
PRONÓSTICO ::= normal | lesión,
ANTECEDENTE ::= EXP
EXP ::= (EXP OPR EXP) | (not (EXP)) | E,
OPR ::= or | and,
OPR2 ::= < | > | ≤ | ≥ ,
E ::= (CLÁUSULA OPR2 real) ,
CLÁUSULA ::= secDifParMax | secDifAngParMax | secDifParMin | secDifAngParMin | parMax | angParMax | timParMax | parMin | angParMin | timParMin | timMedParMaxExt | desTipTimMaxExt | timMedParMinFlx | desTipTimMinFlx | parMedExt | desTipParExt | parMedFlx | desTipParFlx }

Fig. 8.4. Gramática G_{RBS} usada por el generador automático de RBSs para la detección de lesiones de rodilla.

8.3.2 Resultados

Bajo la configuración descrita, se han realizado 100 ejecuciones del generador automático de RBSs mediante PGGG. La Figura 8.5 muestra los resultados medios derivados de la aplicación de los 100 RBSs generados a la detección de lesiones a partir del análisis de curvas isocinéticas.

RBS	Número de reglas	Número de aciertos en el entrenamiento	Número de aciertos en las pruebas	Adaptación en el entrenamiento	Adaptación en las pruebas
Media	56.41	59	14.77	0.8125	0.7385
Desviación Típica	26.3059	3.2906	1.2379	0.0441	0.0618
Tamaño del conjunto de entrenamiento: 72 Tamaño del conjunto de pruebas: 20					

Fig. 8.5. Promedio de resultados obtenidos por 100 bases de reglas construidas por el generador automático de RBSs mediante PGGG.

El valor medio del grado de adaptación en la fase de pruebas decae menos de 0.08 puntos con respecto a la fase de aprendizaje, lo que se traduce en un descenso del porcentaje de aciertos menor al 8%. Se puede concluir que los RBSs resultantes generalizan bien el conocimiento aprendido en el entrenamiento sobre el conjunto de pruebas.

El tamaño medio de las bases de reglas generadas es grande, ya que están conformadas por un número elevado de reglas. Este hecho compensa la corta longitud de las reglas obtenidas, que cuentan en su mayoría con menos de tres asignaciones en el antecedente. La Figura 8.6 contiene una base de reglas típica que ejemplifica la estructura estándar de las bases de reglas obtenidas.

```
if ( parMax <= 129.2357) then ( pronóstico = normal ) ;
if ( parMedExt > 105.4689 ) then ( pronóstico = lesión ) ;
if ( parMax < 102.1192 ) then ( pronóstico = lesión ) ;
if ( timMedParMinFlx > 105.6267 ) then ( pronóstico = lesión ) ;
if ( secDifAngParMax >= -53.1183 ) then ( pronóstico = normal ) ;
if ( desTipTimMaxExt > 41.2201 ) then ( pronóstico = lesión ) ;
if ( ( secDifParMin > -28.7950) and ( secDifAngParMax <= -8.97097824155711 ) )
    then ( pronóstico = lesión ) ;
if ( parMedFlx >= 322.6846 ) then ( pronóstico = lesión ) ;
if ( parMedExt <= 90.2729 ) then ( pronóstico = lesión ) ;
if ( desTipTimMaxExt < 113.8838 ) then ( pronóstico = normal ) ;
if ( timMedParMaxExt <= 112.9948 ) then ( pronóstico = normal ) ;
if ( secDifParMax < 2.4264 ) then ( pronóstico = lesión ) ;
if ( timMedParMaxExt <= 141.6265 ) then ( pronóstico = lesión ) ;
if ( desTipParFlx <= 242.394 ) then ( pronóstico = lesión ) ;
if ( angParMin < 260.8198 ) then ( pronóstico = normal ) ;
if ( angParMax < 547.7457) then ( pronóstico = normal ) ;
if ( secDifAngParMax > 12.5504 ) then ( pronóstico = lesión ) ;
if ( not ( ( secDifAngParMin <= -1.2004) ) ) then ( pronóstico = lesión ) ;
if ( not ( ( timMedParMinFlx > 99.3448) ) ) then ( pronóstico = normal ) ;
if ( not ( ( ( ( timMedParMinFlx <= 39.1191 ) or ( ( ( timMedParMinFlx < 104.1837 ) or (
secDifAngParMax >= 10.1352 ) ) or ( ( secDifAngParMax < 63.3456 ) or ( parMax > 57.7534 ) ) ) ) )
and ( ( ( timMedParMinFlx > 77.4986 ) or ( ( timParMin > 587.8703 ) and ( desTipTimMaxExt <=
24.07503) ) ) or ( desTipTimMinFlx < -64.9569 ) ) ) ) )
    then ( pronóstico = normal ) ;
if ( angParMax <= 510.8999 ) then ( pronóstico = normal ) ;
if ( angParMax <= 523.02476) then ( pronóstico = normal ) ;
if ( ( ( angParMax <= 573.6751 ) and ( parMin < -135.4913 ) ) or ( desTipTimMaxExt > 60.7051 ) )
    then ( pronóstico = lesión ) ;
if ( parMax <= 128.0009 ) then ( pronóstico = normal ) ;
if ( ( angParMax >= 550.5783 ) and ( desTipTimMinFlx <= -43.142 ) )
    then ( pronóstico = lesión ) ;
```

Fig. 8.6. Base de reglas típica.

8.4 Generación automática de un FRBS para detectar lesiones de rodilla a través de FNNs mediante PGGG

8.4.1 Configuración específica

Las características descritas en la Figura 8.2 son las 18 entradas reales al FRBS. Para cada una de ellas se han creado tres conjuntos difusos que corresponden a las etiquetas lingüísticas: "bajo", "medio" y "alto". Esta tarea se ha llevado a cabo mediante el algoritmo de clustering Fuzzy C-Means (FCM). La variable de salida real "pronóstico" tiene asociados dos conjuntos difusos cuyas etiquetas lingüísticas son "normal" y "lesión", respectivamente. Atendiendo a esta configuración, la longitud de un individuo de la población del PGGG es la siguiente:

$$(I \times C + O \times K) \times R = (18 \times 3 + 1 \times 2) \times R = 56 \times R$$

siendo I, O el número de variables lingüísticas de entrada y salida respectivamente; C, el número de conjuntos difusos asociados a las variables de entrada; K, el número de conjuntos difusos asociados a las variables de salida y R el número de reglas difusas.

La gramática G_{FNN} mostrada en la Figura 8.7 genera un lenguaje cuyas palabras son árboles de derivación que codifican individuos de estas características. El motor de inferencia utiliza la t-norma del mínimo para los operadores difusos de conjunción e implicación. De los dos métodos de desborrosificación presentados, se ha seleccionado el método FATI y el operador de agregación elegido (disyunción) ha sido la t-conorma del máximo.

$G_{FNN} = (\Sigma_N, \Sigma_T, S, P)$

$\sum_N$ = { S, I, H, A, B, C, Z }

$\sum_T$ = { 0, 1 }

P = { S::= I,

I ::= I H | H,

H ::= A Z,

A ::= C B B B B B B B B B B B B B B B B B | B C B B B B B B B B B B B B
B B B B

B B C B B B B B B B B B B B B B B B | B B B C B B B B B B B B B B B
B B B

B B B B C B B B B B B B B B B B B B | B B B B B C B B B B B B B B B
B B B

B B B B B B C B B B B B B B B B B B | B B B B B B B C B B B B B B B
B B B

B B B B B B B B C B B B B B B B B B | B B B B B B B B B C B B B B B
B B B

B B B B B B B B B B C B B B B B B B | B B B B B B B B B B B C B B B
B B B

B B B B B B B B B B B B C B B B B B | B B B B B B B B B B B B B C B
B B B

B B B B B B B B B B B B B B C B B B | B B B B B B B B B B B B B B B
C B B

B B B B B B B B B B B B B B B B C B | B B B B B B B B B B B B B B B
B B C,

B ::= 0 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1 0 0 | 1 0 1 | 1 1 0,

C ::= 0 0 1 | 0 1 0 | 0 1 1 | 1 0 0 | 1 0 1 | 1 1 0,

Z ::= 1 0 | 0 1 }

Fig. 8.7. Gramática G_{FNN} usada por el generador automático de FRBSs para la detección de lesiones de rodilla.

8.4.2 Resultados

Con estos parámetros, se han realizado 100 ejecuciones del generador automático de FRBSs a través de FNNs mediante PGGG. La Figura 8.8 muestra los resultados derivados de la aplicación de los 100 FRBSs generados a la detección de lesiones a partir del análisis de curvas isocinéticas.

FNN	Número de reglas	Número de aciertos en el entrenamiento	Número de aciertos en las pruebas	Adaptación en el entrenamiento	Adaptación en las pruebas
Media	8.42	66.56	14.99	0.9152	0.7495
Desviación Típica	4.1637	2.8438	1.3669	0.0403	0.0683
Tamaño del conjunto de entrenamiento: 72 Tamaño del conjunto de pruebas: 20					

Fig. 8.8. Promedio de resultados obtenidos por 100 bases de reglas construidas por el generador automático de FNNs mediante PGGG.

El número de aciertos obtenido en el conjunto de curvas isocinéticas de entrenamiento es muy alto, lo cual muestra la alta capacidad de adaptación de las bases de reglas difusas al conjunto de datos. El grado de adaptación medio de la fase de pruebas refleja un porcentaje de aciertos cercano al 75%, ligeramente mayor que la media obtenida por los RBSs. Este dato es alcanzado con FNNs que codifican bases de un tamaño medio aproximado de 8 reglas difusas. Por el contrario, los antecedentes de estas reglas asignan valores a casi todas las variables de entrada al FRBS, dando lugar a reglas extensas. La Figura 8.9 muestra una base de reglas difusas representativa del conjunto resultante del generador automático.

if (SecDifParMax is medio or alto) and (ParMax is alto) and (TimParMax is bajo or medio) and (ParMin is medio or alto) and (TimMedParMaxExt is bajo or alto) and (DesTipParExt is bajo or alto) and (DesTipParFlx is bajo or alto)
then Pronóstico is normal;
if (SecDifAngParMin is medio or alto) and (AngParMax is bajo or alto) and (TimParMax is bajo or alto) and (ParMin is bajo or alto) and (ParMedFlx is bajo or alto) and (DesTipParExt is bajo) and (DesTipParFlx is bajo or alto)
then Pronóstico is normal;
if (SecDifAngParMax is bajo or alto) and (SecDifAngParMin is bajo or alto) and (TimParMax is bajo or alto) and (TimMedParMaxExt is medio or alto) and (DesTipTimMaxExt is bajo or alto) and (DesTipParExt is bajo or alto) and (DesTipParFlx is bajo or alto)
then Pronóstico is normal;
if (ParMax is bajo or alto) and (TimParMax is bajo or alto) and (ParMin is bajo or alto) and (AngParMin is bajo) and (TimMedParMinFlx is alto) and (ParMedExt is medio or alto) and (DesTipTimMaxExt is bajo or alto) and (DesTipParExt is bajo or alto)
then Pronóstico is lesión;
if (SecDifParMax is bajo or alto) and (SecDifParMin is bajo or alto) and (TimParMin is bajo or alto) and (TimMedParMaxExt is bajo or alto) and (ParMedFlx is bajo or alto) and (DesTipTimMinFlx is bajo or alto) and (DesTipParFlx is bajo or alto)
then Pronóstico is lesión;

Fig. 8.9. Base de reglas difusas prototípica.

8.5 Discusión

A continuación se presenta una comparativa de los resultados obtenidos por ambas técnicas con el fin de evaluar cuantitativa y cualitativamente sus ventajas e inconvenientes. Se ha realizado un test ANOVA para comparar los valores medios del grado de adaptación, tanto en la fase de entrenamiento como en la de pruebas, de ambos sistemas basados en reglas.

Se han planteado dos hipótesis nulas H_0^1 y H_0^2:

1. H_0^1: El grado de adaptación medio en la fase de entrenamiento de los RBSs es igual al grado de adaptación medio en la fase de entrenamiento de los FRBSs.

2. H_0^2: El grado de adaptación medio en la fase de pruebas de los RBSs es igual al grado de adaptación medio en la fase de pruebas de los FRBSs.

Tal y como se muestra en la Figura 8.10, los grados de adaptación medios de entrenamiento y de pruebas de ambas aproximaciones son diferentes, siendo la diferencia más acusada entre los grados de adaptación de entrenamiento. El valor máximo del grado de adaptación de entrenamiento es alcanzado por un FRBS, así como el valor máximo del grado de adaptación de pruebas. Los valores más pequeños del grado de adaptación corresponden en ambos casos a RBSs. Por tanto, los datos indican a priori que los FRBSs detectan lesiones de rodilla de manera más precisa que los RBSs.

ESTADÍSTICA DESCRIPTIVA								
					95% Intervalo de Conf. para la Media			
	Nº	Media	Des. Típica	Error Típico	Límite inferior	Lím. superior	Mínimo	Máximo
Adapt. Entrenam.								
RBS	100	,8125	,04412	,00441	,8037	,8213	,69	,92
FRBS	100	,9152	,04034	,00403	,9072	,9232	,80	1,00
Total	200	,8639	,06654	,00471	,8546	,8731	,69	1,00
Adapt. Pruebas								
RBS	100	,7385	,06190	,00619	,7262	,7508	,55	,85
FRBS	100	,7495	,06835	,00683	,7359	,7631	,60	,90
Total	200	,7440	,06527	,00462	,7349	,7531	,55	,90

Fig. 8.10. Estadísticas descriptivas de las distribuciones de valores del grado de adaptación para ambas aproximaciones durante las fases de entrenamiento y pruebas.

En la Figura 8.11 se recogen los resultados del test de Levene o test de homogeneidad de las varianzas (homocedasticidad), realizado sobre las distribuciones de los grados de adaptación de entrenamiento y pruebas de ambas técnicas. Este test prueba

estadísticamente que ambas distribuciones son homocedásticas, requisito para que los resultados de ANOVA sean válidos. Dadas las hipótesis nulas de que las varianzas de ambas distribuciones son homocedásticas, los resultados prueban que no se pueden rechazar dichas hipótesis dado $p_{entrenamiento} < 0.455$ y $p_{prueba\ s} < 0.581$.

TEST DE HOMOGENEIDAD DE LAS VARIANZAS				
	Levene	df1	df2	Sig.
Adaptación Entrenamiento	,561	1	198	,455
Adaptación Pruebas	,306	1	198	,581

Fig. 8.11. Resultados del **t**est de homogeneidad de las varianzas.

La Figura 8.12 muestra los resultados del la aplicación de ANOVA. A partir de ellos se rechaza la hipótesis nula H_0^1 dado $p < 0.01$. Estos resultados indican que los grados de adaptación medios de la fase de entrenamiento son diferentes en ambas técnicas, por lo que se puede afirmar que los FRBSs aprenden mejor el conjunto de datos de entrenamiento que los RBSs. Por el contrario, la hipótesis nula H_0^2 no puede ser rechazada ($p < 0.234$), lo que demuestra estadísticamente que ambos sistemas son igual de eficientes detectando lesiones de rodilla en la fase de pruebas.

ANOVA						
		Suma de los cuadrados	df	Media cuadrática	F	Sig.
Adaptación Entrenamiento	Inter-grupos	,527	1	,527	295,158	,000
	Intra-grupos	,354	198	,002		
	Total	,881	199			
Adaptación Pruebas	Inter-grupos	,006	1	,006	1,423	,234
	Intra-grupos	,842	198	,004		
	Total	,848	199			

Fig. 8.12. Resultados del **t**est ANOVA.

La comprensión del conocimiento almacenado en las bases de reglas es otro aspecto relevante a valorar, ya que la siguiente etapa en la creación de los sistemas generados es el refinamiento de sus reglas por parte del experto. Las bases de reglas de los RBSs están compuestas por muchas reglas de escasa longitud, en contraposición a las de los

FRBSs, que contienen pocas reglas con antecedentes muy completos. Las reglas cortas de un RBS son sencillas de comprender individualmente, lo que provee al experto con conocimiento individualizado para cada variable de entrada, pero el entendimiento de la totalidad de la base se hace difícil de abordar por la abundancia de reglas. El análisis aislado de las reglas más extensas no es tan sencillo, debido al uso reiterado de la negación y a la abundancia de expresiones anidadas entre paréntesis que dificultan la lectura. Afortunadamente, la probabilidad de aparición de este tipo de reglas es baja. Las reglas difusas de los FRBSs son extensas, sin embargo, la codificación intermedia en FNNs les confiere una estructura ordenada que facilita su entendimiento. Además, las reglas difusas están compuestas íntegramente por términos lingüísticos, lo que las hace próximas a la manera de expresarse del experto.

En conclusión, ambas aproximaciones han demostrado su capacidad para extraer conocimiento de los datos de entrenamiento y aplicarlo satisfactoriamente a la detección de lesiones en pacientes que no han sido presentados anteriormente. En el caso de los RBSs, la representación del conocimiento extraído es, en líneas generales, menos accesible por parte del experto que en los FRBSs, pero permite realizar un análisis individualizado de cada variable de entrada al sistema. Un FRBS representa el conocimiento de manera más intuitiva, facilitando la comprensión global de toda la base de reglas y aportando una visión sobre el dominio complementaria a la proporcionada por un RBS.

9. Conclusiones y líneas futuras

9.1 Conclusiones

En este trabajo se ha propuesto un sistema automático generador de bases de conocimiento basadas en reglas. Este sistema reduce la dependencia del experto del dominio existente en el proceso de creación de una base de conocimiento, lo que disminuye el coste derivado del mismo y alivia el cuello de botella generado por las técnicas tradicionales de educción de conocimientos.

El sistema propuesto resuelve el problema de búsqueda y optimización de bases de conocimiento basadas en reglas mediante programación genética guiada por gramáticas. Para ello, parte de un modelo de representación sustentado en la teoría de los lenguajes formales, que permite definir una gramática formal que genera un lenguaje cuyas palabras codifican bases de conocimiento basadas en reglas. La programación genética guiada por gramáticas solventa parte de los inconvenientes presentados por otros algoritmos de computación evolutiva, como los algoritmos genéticos y la programación genética. En contraposición a los métodos de codificación de individuos de los algoritmos genéticos, la codificación de individuos en la programación genética guiada por gramáticas no establece un tamaño máximo para los mismos, permitiendo codificar bases de conocimiento con un número variable de reglas. A diferencia de la programación genética, el cruce, la mutación y la generación de la población inicial en la programación genética guiada por gramáticas se llevan a cabo de acuerdo a una gramática formal, lo que impide la generación individuos que codifiquen bases de conocimiento no válidas, traduciéndose en una mejora de la velocidad de convergencia y una disminución de la probabilidad de caída en óptimos locales.

Se han presentado dos variaciones del sistema: la primera genera directamente sistemas basados en reglas (RBS) y la segunda genera indirectamente sistemas basados en reglas difusas (FRBS) representados a través de redes de neuronas difusas (FNN). Para ambas aproximaciones se ha diseñado un sistema de codificación de individuos específico, una

gramática libre de contexto que permite la generación de individuos sujetos a dicha codificación y un método de evaluación de individuos especializado para el propósito del dominio de aplicación del sistema. El dominio de aplicación propuesto es la detección de lesiones de rodilla a partir del análisis de curvas isocinéticas.

Los buenos resultados obtenidos muestran la eficiencia de ambas aproximaciones al clasificar curvas isocinéticas en aquellas pertenecientes a un paciente lesionado en la rodilla y a uno sano, además de proporcionar bases de conocimiento que muestran al experto del dominio el conocimiento extraído de los datos de manera comprensible. Otra aportación importante del sistema es el método de análisis llevado a cabo sobre las curvas isocinéticas. Una curva isocinética es una serie temporal de longitud variable que, mediante este método de análisis, es traducida en un vector de características de dimensión finita procesable por el generador automático de sistemas basados en reglas mediante programación genética guiada por gramáticas.

La codificación directa de RBSs da lugar a bases de conocimiento con un alto número de reglas, en contraposición a las bases de reglas difusas codificadas a través de FNNs, que son de menor tamaño pero constan de reglas de mayor longitud. Estas dos aproximaciones proporcionan dos visiones complementarias del conocimiento de un mismo dominio, reflejando la capacidad del generador automático mediante programación genética guiada por gramáticas de extraer grandes cantidades de conocimiento de forma automática. Cabe destacar la robustez de los sistemas, que pueden adaptarse automáticamente a los cambios que acontecen en el dominio para el que han sido desarrollados, facilitando su proceso de mantenimiento con actualizaciones constantes sencillas de realizar. Este hecho supone un avance dentro de la línea de investigación en inteligencia artificial encaminada a la construcción de sistemas inteligentes robustos. Atendiendo a los resultados obtenidos, se puede afirmar que los RBSs y FRBSs generados automáticamente mediante programación genética guiada por gramáticas, sirven como sistemas de segunda opinión y sistemas inteligentes para la ayuda a la toma de decisiones al doctor encargado de detectar de lesiones de rodilla.

Resultados parciales de este trabajo han sido enviados para su defensa y publicación en el Congreso Europeo de Aprendizaje Automático y Prácticas de Adquisición de Conocimiento de Bases de Datos (European Conference on Machine Learning and Practice of Knowledge Discovery in Databases, ECML PKDD), que se celebra en Amberes el 19 de septiembre de 2008. Este congreso ocupa la posición 24 de 620 en el índice CS Conference Rankings (http://www.cs-conference-ranking.org/conference rankings/alltopics.html). Esta publicación internacional se considera de reconocido prestigio según se indica en la resolución del 6 de noviembre de 2007 del Ministerio de Educación y Ciencia, recogida en el BOE del 21 de noviembre de 2007.

9.2 Líneas futuras

A partir de los trabajos y resultados de investigación de esta tesis de fin de máster, es posible realizar sistemas generadores de bases de reglas difusas que evolucionen poblaciones de individuos capaces de codificar las funciones de pertenencia a los conjuntos difusos de la base de hechos. De esta forma, el espacio de soluciones recorrido por el programa genético aumenta, pero también se incrementa la flexibilidad y la potencia de éste, ya que evoluciona bases de conocimiento completas, generando la estructura de la base de reglas a la par que las formas de los conjuntos difusos de las variables lingüísticas de entrada y salida del sistema.

En relación a la velocidad de convergencia del programa genético, se debe tener en cuenta la utilización de gramáticas libres de contexto estocásticas. Estas gramáticas son una extensión de las gramáticas libres de contexto, a cuyas producciones les son asignados valores probabilísticos que determinan su probabilidad de elección durante la ejecución del algoritmo de generación de individuos y de los operadores de cruce y de mutación. Con el estudio de esta técnica, parece posible evaluar la relevancia de las producciones de la gramática, asignándoles un valor probabilístico capaz de guiar al programa genético para que genere árboles de derivación usando las producciones mejor valoradas de la gramática. Es posible incluso evolucionar dichas probabilidades paralelamente a la población de individuos del programa genético, de tal forma que la gramática libre de contexto estocástica fuese refinándose generación tras generación. De

esta línea de investigación pueden surgir técnicas que ayuden a facilitar la convergencia de un programa genético e incluso que dificulten su caída en óptimos locales.

Una de las ventajas del generador automático mediante PGGG es que puede extenderse a la construcción de sistemas basados en el conocimiento que hagan uso de otras representaciones del mismo. Un interesante punto de partida es el estudio de la construcción de redes bayesianas mediante PGGG, dando lugar a la generación de una gramática libre de contexto capaz de codificar topologías bayesianas como árboles de derivación. Adicionalmente, se puede considerar la posibilidad de codificar, no únicamente una topología de red, sino una red bayesiana completa incluyendo las probabilidades condicionadas de sus nodos como parte de los individuos de la población, de tal forma que el programa genético fuese capaz de generar una red bayesiana completa para su aplicación en un dominio específico.

10. Bibliografía

[ANIB04] Aníbal, L.M., Martin, J., Lorenz, R.D.: "Expert System for Integrated Control and Supervision of Dry-End Sections of Paper Machines." IEEE Transactions on Industry Applicactions, Vol. 40, no. 2, pp 680-691, 2004.

[ATMA76] Atmar, J.W.: "Speculation of the Evolution of Intelligence and its Possible Realization in Machine Form". Tesis Doctoral, New Mexico State University, 1976.

[BAKE87] Baker, J.E.: "Reducing Bias and Inefficiency in the Selection Algorithm". Proceedings of the 1st International Conference on Genetic Algorithms, pp 101-111, 1987.

[BARR91] Barrios, D.: "Operador de Cruce Generalizado en Algoritmos Genéticos". Tesis Doctoral, Facultad de Informática, Universidad Politécnica de Madrid, 1991.

[BREM62] Bremermann, M.F.: "Optimization through Evolution and Recombination". Self Organizing Systems, 1962.

[BREM65] Bremermann, M.F., Rogson, M., and Salaff, S.: "Search by Evolution Biophysics and Cybernetic Systems". Proceedings of the 2nd Cybernetic Sciences Symposium, pp 157-167, 1965.

[BRIN91] Brindle, A.: "Genetic Algorithms for Function Optimization". Tesis Doctoral, University of Alberta, 1991.

[BOX57] Box, G.E.P.: "Evolutionary Operation: A Method for Increasing Industrial Productivity". Applications on Statistics (6), pp 81-101, 1957.

[BOX69] Box, G.E.P. and Draper, N.P.: "Evolutionary Operation: A Method for Increasing Industrial Productivity". Willey, 1969.

[CARR03] Carrascal, A., Manrique, D., Ríos, J, Rossi, C.: "Evolutionary Local Search of Fuzzy Rules through a Novel Neuro-Fuzzy Encoding Method". Evolutionary Computation, vol. 11, nº 4, pp 439-461, 2003.

[CHAN95] Chan, Y.F., Ma, H.K., Chan, F.T., Chen, H.Y., Chen, T.Y.: "Teaching Family Planning with Expert System." Computers Educ., Vol. 24, no.4, pp.293-298, 1995.

[CORD97] Cordón O., Herrera, F., Peregrin, A.: "Applicability of the Fuzzy Operators in the Design of Fuzzy Logic Controllers." Fuzzy Sets and Systems 86, pp 15-41, 1997.

[CORD01] Cordón, O., Herrera, F., Hoffmann, F., Magdalena, L.: "Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases." World Scientific, 2001.

[COUC06] Couchet, J., Manrique, D., Ríos, J., Rodríguez-Patón, A.: "Crossover Operators for Grammar-Guided Genetic Programming." Soft Computing: A Fusion of Foundations, Methodologies and Applicactions 11, no. 10, pp 943-955, 2006.

[COUC07] Couchet, J., Manrique, D., Porras, L.: "Grammar-Guided Neural Architecture Evolution". IWINAC, vol. 4527, pp 437-446, 2007.

[CHOM57] Chomsky, N.: "Syntactic Structures". Mouton, 1957.

[CROW70] Crow J.F., Kimura, M.: "An Introduction to Population Genetics Theory". New York. Harper and Row, 1970.

[DARW59] Darwin, C.R.: "On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life", Murray, 1859.

[DAVI91] Davis, L.: "The Handbook of Genetic Algorithms". Van Nostrand Reinhold, New York, 1991.

[DEJO75] De Jong, K.A.: "Analysis of Behaviour of a class of Genetic Adaptive Systems". Tesis Doctoral, University of Michigan, 1975.

[DORA99] Dorado de la Calle, J: "Modelo de un Sistema para la Selección Automática en Dominios Complejos, con una Estrategia Cooperativa, de Conjuntos de entrenamiento y Arquitecturas Ideales de Redes de Neuronas Artificiales utilizando Algoritmos Genéticos". Tesis Doctoral. Universidad de La Coruña, 1999.

[FOGE62] Fogel, L.J.: "Autonomous Automata Industrial". Res. 4, 1962.

[FOGE66] Fogel, L.J., Owens, A.J. and Walsh, M.J.: "Artificial Intelligence through Simulated Evolution". Wiley, 1966.

[FRAS57] Fraser, A.S.: "Simulation of Genetic Systems by Automatic Digital Computers". Australian Journal Biological Sciences (10), pp 484-499, 1957.

[FRIE58] Friedberg, R.M.: "A learning Machine: Part I". IBM Journal (2), pp 2-13, 1958.

[FRIE59] Friedberg, R.M., Dunham, B. and North, J.H.: "A learning Machine: Part II". IBM Journal (3), pp 282-287, 1959.

[GARC07] García-Arnau, M., Manrique, D., Ríos, J., Rodríguez-Patón, A.: "Initialization method for grammar-guided genetic programming". Knowledge-Based Systems, vol. 20, nº 10, pp 127-133, 2007.

[GOET95] Goethe, J.W., Bronzino, J.D.: "An Expert System for Monitoring Psychiatric Treatment." IEEE Engineering in Medicine and Biology Magazine, Vol. 14, no. 6, pp 776-780, 1995.

[GOLD89] Goldberg, D.E.: "Zen and the Art of Genetic Algorithms". Proceedings of the 3rd International Conference on Genetic Algorithms, pp 80-85, 1989.

[GOLD91] Goldberg, D.E., Deb, K.: "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms". Foundations of Genetic Algorithms, pp 69-93, 1991.

[GOME97] Gomez, A., Juristo, N., Montes, C., Pazos, J.: "Ingeniería del Conocimiento." Centro de Estudios Ramón Areces, 1997.

[GROS04] Grosman, B., Lewin, D.R.: "Adaptive Genetic Programming for Steady-State Process Modeling". Computers and Chemical Engineering, (28), pp 2779-2790, 2004.

[HOAI02] Hoai, N.X., McKay, R.I.: "Is Ambiguity Useful or Problematic for Grammar Guided Genetic Programming? A case of Study". Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning, Singapore, pp 449-453, 2002.

[HOFF89] Hoffmann, A.: "Arguments on Evolution: A Paleontologist's Perspectiva". Oxford University Press, New York, 1989.

[HOLL69] Holland, J.H. "Adaptative Plans Optimal for Playoff-Only Environments". Proceedings of the 2nd Hawai International Conference on System Sciences, pp 917-920, 1969.

[HOLL75] Holland, J.H. "Adaptation in Natural and Artificial Systems". University of Michigan, Ann Arbor, 1975.

[HOPC02] Hopcroft, John E.: "Introducción a la teoría de autómatas, lenguajes y computación". Pearson Educación, 2002.

[HUSS03] Hussain, T.S.: "Attribute Grammar Encoding of the Structure and Behaviour of Artificial Neural Networks". Tesis Doctoral, Queen's University. Kingston, Ontario, Canada., 2003.

[ISAS97] Isasi, P., Martínez P., Borrajo, D.: "Lenguajes, Gramáticas y Autómatas. Un enfoque práctico". Addison-Wesley, 1997.

[KARP87] Karp, R.M., Rabin. M.: "Efficient randomized pattern-matching algorithms". IBM Journal of Research and Development, 31(2), pp 249-260, 1987.

[KOZA92] Koza, J.R.: "Genetic Programming: On the Programming of Computers by Means of Natural Selection". MIT Press, Cambridge, MA, 1992.

[LANG87] Langton, C.G.: "Artificial Life". Addison-Wesley, 1987.

[LARO06] larosa, D. T.: "Data Mining: Methods and Models". Wiley, 2006.

[MAHA03] Mahaman, B.D., Passam, H.C., Sideridis, A.B., Yialouris, C.P.: "DIARES-IPM: a Diagnostic Advisory Rule-Based Expert System for Integrated Pest Management in Solanaceous Crop Systems." Agricultural Systems 76, pp 1119-1135, 2003.

[MANR05] Manrique, D., Márquez, A., Ríos, J., Rodríguez-Patón, A.: "Grammar based crossover operator in Genetic Programming". Lecture notes in computer Science, Vol. 3562: 252-261, 2005.

[MANR06] Manrique, D., Ríos, J., Rodríguez-Patón, A.: "Evolutionary System for Automatically Constructing and Adapting Radial Basis Function Networks." NeuroComputing 66, no. 16-18, pp 2268-2283, 2006.

[MAYR88] Mayr, E.: "Toward a New Philosophy of Biology: Observations o fan Evolutionist". Belknap, 1988.

[NEUM56] Von Neumman, J.: "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components". Automata Studies (C.E. Shannon and J. McCarthy,eds). Princeton University Press, 1956.

[NEUM58] Von Neumman, J.: "The Computer and the Brain". Yale University Press, 1958.

[NGUY97] Nguyan, H.T., Walker, E.A.: "A First Course in Fuzzy Logic." CRC Press, 1997.

[PAJA05] Pajares, G., Santos, M.: "Inteligencia Artificial e Ingeniería del Conocimiento." Ra-Ma, 2005.

[PANA04] Panait, L., Luke, S: "Alternative Bloat Control Methods". GECCO, vol.3103, pp 630-641, 2004.

[RAHM96] Rahman, S., Hazim, O.: "Load Forecasting for Multiple Sites: Development of an Expert System-Based Technique." Electric Power Systems Research 39, pp 161-169, 1996.

[RAY91] Ray, T.: "An Approach to the Synthesis of Artificial Life II". Addison-Wesley, 1991.

[RECH65] Rechenberg, I.: "Cybernetic Solution Path of an Experimental Problem Royal Aircraft". Establishment Library Translation, 1965.

[RECH94] Rechenberg, I.: "Evolution Strategies '94". Frommann-Holzboorg, 1994.

[RODR02] Rodrigues, E., Pozo, A.: "Grammar-Guided Genetic Programming and Automatically Defined Functions". Proceedings of the 16th Brazilian Symposium on Artificial Intelligence. Recife, Brasil, pp 324-333, 2002.

[SPEN62] Spendley, W., Hext, G.R. and Himsworth, F.R.: "Sequential Application of Simplex Designs in Optimization and Evolutionary Operation". Technometrics (4), pp 441-461, 1962.

[SCHW91] Schwefel, H.P. and Männer, R.: "Parallel Problem Solving from Nature". Proceedings of the 1st Workshop PPSN I. Springer, Berlin, pp 307 – 313, 1991.

[SCHW95] Schwefel, H.P.: "Evolution and Optimum Seeking". Wiley, New York, 1995.

[TRIL92] Trillas, E., et al.: "Aplicaciones de la Lógica Borrosa." Consejo Superior de Investigaciones Científicas, 1992.

[WHIG95] Whigham, P.A.: "Grammatically-Based Genetic Programming". Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications, California, USA, pp 33-41, 1995.

[WHIG96] Whigham, P.A.: "Grammatical Bias for Evolutionary Learning", Tesis Doctoral, School of Computer Science, Australian Defence Force (ADFA), University College, University of New South Wales, Australia, 1996.

[WINS92] Winston, P.H.: "Artificial Intelligence Third Edition." Addison-Wesley Publishing Co., 1992.

[WITT05] Witten, I. H., Frank, E.: "Data Mining: Practical Machine Learning Tools and Techniques". Elsevier, 2005.

[WOLP96] Wolpert, D.H. and Macready, W.G.: "No Free Lunch Theorems for Search". Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1996.

[ZADE65] Zadeh, L.A.: "Fuzzy Sets." Information and Control 8, pp 338-353, 1965.

[ZAHE06] Zaheeruddinm, Garima: "A Neuro-Fuzzy Approach for Prediction of Human Work Efficiency in Noisy Environment". Applied Soft Computing, vol. 6, pp 283-294, 2006.