VERIFICACIÓN DE ALGORITMOS DE CLASIFICACIÓN CON PRECONDICIÓN DIFUSA

Victoria López
Dept. Estadística e I.O. I
Universidad Complutense
vlopez@mat.ucm.es

Daniel Gómez
Dept. Estadística e I.O. III
Universidad Complutense
dagomez@estad.ucm.es

Javier Montero
Dept. Estadística e I.O. I
Universidad Complutense
javier_montero@mat.ucm.es

Resumen

En este artículo abordaremos el problema de la verificación formal [7, 13] de un algoritmo con precondiciones y entradas difusas. Se tratará la verificación mediante la semántica axiomática realizando transformaciones de predicados difusos. Para ello comenzaremos presentando la especificación formal de algoritmos nítidos que ampliaremos al contexto difuso para dar paso a la metodología de verificación formal nítida y difusa. Presentaremos un problema de decisión-clasificación para ejemplificar el objetivo de este artículo.

Palabras Clave: Verificación, Especificación, Algoritmos Difusos.

1 INTRODUCCIÓN

En este trabajo se presenta una metodología para la verficación de algoritmos de clasificación difusa de imágenes digitales obtenidas vía satélite (teledetección).

Los problemas de clasificación de imágenes vía satélite han sido ampliamente estudiados en la literatura. Durante las dos últimas décadas han aparecido numerosas técnicas para el reconocimiento de patrones y clasificación para la extracción de información de datos procedentes de teledetección (ver [10]).

Actualmente la clasificación multiespectral puede realizarse utilizando una gran variedad de algoritmos, que se pueden clasificar en dos grandes grupos: clasificación nítida y clasificación utilizando conjuntos difusos.

Debido a la naturaleza de las imágenes así como el proceso mediante el cuál se obtienen es deseable abordar el problema de clasificación desde el contexto difuso.

En la **clasificación supervisada**, la identidad, localización y características de algunos tipos de terreno son conocidos a priori, mediante análisis de fotografías aéreas, mapas y experiencia personal. El analista intenta encontrar lugares específicos en los datos que representen ejemplos de estos tipos de terreno conocidos. Estas áreas se conocen como *training sites*, ya que las características de estos tipos de area se suelen determinar mediante algoritmos de aprendizaje. Cada pixel es evaluado y será asignado a la clase a la cual tiene mayor probabilidad de pertenecer (en el contexto difuso se obtendrán funciones de pertenencia para cada una de las clases).

En la **clasificación no supervisada**, las clases, así como otras características, no son conocidas a priori. En estos casos se ha de agrupar los pixels con características similares dentro de un cluster de acuerdo con algún criterio determinado (ver Amo et al [1, 2]). El algoritmo que se evalúa en este trabajo es proceso de una larga investigación en clasificación no supervisada para la determinación de regiones homogéneas y posterior clasificación difusa (ver [1, 2, 3, 4]).

Uno de los principales problemas que surge de forma natural en cualquier algoritmo de clasificación no supervisado (tanto en el caso nítido como en el difuso) es el de la validación del proceso de clasificación. En la mayoría de las ocasiones este proceso de evaluación es hecho de forma empírica por expertos que deciden si la mayoría de las clasificaciones que se realizan son adecuadas o no lo son.

Sin embargo este método de evaluación tiene ciertos problemas:

- En algunos campos (problemas de optimización de rutas) existen problemas test para determinar las bondades de un algoritmo, esto puede hacerse porque tienen una función objetivo clara-

mente definida. En problemas de clasificación de imágenes vía satélite no existe esta colección de problemas test y por tanto hace más difícil una valoración desde un punto de vista empírico.

- Frente a los mismos resultados pueden existir interpretaciones y validaciones distintas (ver [8]) hechas por diferentes expertos (lo que dificulta la valoración del algoritmo por esta vía).

Por este motivo existe una necesidad de realizar una validación del algoritmo de clasificación que tenga en cuenta el proceso y no únicamente el resultado final.

La verificación de algoritmos en el contexto nítido surge como necesidad de demostrar que el algoritmo hace lo que nosotros queremos que realmente haga. En el caso difuso la verficación es un poco más flexible y podría entenderse como una validación y análisis del proceso desarrollado. Además, el uso de la verificación de procesos nítidos como localización de errores y depuración de algoritmos puede aplicarse también en el caso difuso.

Así pues, es necesario desarrollar otras técnicas (como la vericación de algoritmos) con el objeto de validar y analizar procesos de clasificación no supervisada en los que la vía del empirismo no es suficiente.

Este trabajo está dividido de la siguiente forma: en la sección 2 se introduce la especificación formal de algoritmos en el caso nítido y difuso. En la sección 3 se ve la verificación formal para el caso iterativo y se analiza el proceso de verificación de algoritmos difusos. En la sección 4 se aborda el problema de la especificación y diseño de un algoritmo de clasificación-decisión propuesto por los autores en trabajos anteriores [1, 2, 3, 4, 5] y se verifica en la sección 5. Finalmente en la sección 6 se presentan las conclusiones finales.

2 ESPECIFICACIÓN FORMAL DE ALGORITMOS

Todo problema objeto de solución tiene asociados datos e información relativa al problema planteado. Formalizamos la especificación algorítmica como la relación formal, expresada con lógica de primer orden, entre los datos iniciales y los datos finales.

2.1 ESPECIFICACIÓN NÍTIDA

La especificación formal es la parte de la algorítmica que se ocupa de determinar, mediante aserciones lógicas, las condiciones de funcionamiento y el resultado que va a ser calculado por el algoritmo [11]. Consta de tres partes fundamentales:

Cabecera del programa o subprograma donde se especifica el nombre del subprograma, los datos parámetros de entrada y de salida, su modo y su tipo.
Precondición, predicado lógico que representa uno o varios requisitos que deben cumplir los datos de entrada para garantizar una ejecución correcta del algoritmo.
Postcondición, predicado lógico que se hace cierto tras una ejecución correcta del algoritmo, nos informa del valor o valores que tomarán los parámetros de salida en función de los datos de entrada.

2.2 ESPECIFICACIÓN DIFUSA

Mediante este título nos referimos a aquellos algoritmos cuyos parámetros de entrada deben cumplir por precondición una fórmula difusa, es decir que pueda no ser estrictamente cierta o estrictamente falsa, sino que el cumplimiento de la precondición por parte de los datos de entrada sea un grado de adecuación a las restricciones del algoritmo.

Ejemplo 2.1 *Se dice que una matriz de $n \times m$ números enteros es melchoriforme si posee algún elemento 'rubio'. Un elemento es 'rubio' si su valor coincide con la suma de los restantes elementos de la matriz.*
En su versión nítida, obtendríamos la suma de los elementos de la matriz y realizaríamos la búsqueda del elemento descrito. Sin embargo, puede interesarnos conocer si existe algún elemento que sin ser 'rubio' esté cerca de serlo. De esta forma proponemos la versión difusa del problema. En este caso lo primero será proponer una función que defina el significado difuso de 'rubio' asícomo de la función 'suma':

$$suma(m) = \sum_{i=1}^{n} \sum_{j=1}^{m} m[i,j]$$

$$rubio(i,j,m) \Leftrightarrow (suma(m) = 2m[i,j])$$

el predicado 'rdif' representa la versión difusa del predicado 'rubio':

$$rdif(i,j,m) \Leftrightarrow (suma(m) \cong 2m[i,j])$$

La especificación del algoritmo melchoriforme difuso resulta:

fun *Melchorif (m:* **array***[1..n,1..n]* **of int** *):*
ret *b:***bool***;*
$\{PRE_{dif}$*: Cierto*$\}$
$\{POST_{dif} : b \Leftrightarrow \exists(i,j) \in \{1..n\} \times \{1..m\} : rdif(i,j,m)\}$

donde la precondición Cierto indica que los datos de entrada no tienen que cumplir ninguna restricción.

Figura 1: Esquema Iterativo

{PRE}
Inic;
{INV}
while Cond **do**
SecInst;
endwhile
{POST}

Por otra parte, la postcondición será cierta en cierto grado, es decir, el predicado $POST_{dif}$ será 1 si en la matriz existe un elemento (i,j) que cumpla el predicado nítido $rubio(i,j,m)$; $POST_{dif}$ será 0 todos los elementos de la matriz cumplen $rdif(i,j,m) = 0$; en otro caso $POST_{dif}$ tomará un valor en (0,1) que representará el grado de certeza en que la matriz dada cumple la propiedad melchoriforme, que debe corresponder con:

$$max_{1 \leq i \leq n, 1 \leq j \leq m}(suma(m) \cong 2m[i,j])$$

$\diamond$

3 VERIFICACIÓN FORMAL

La verificación de un algoritmo consiste en demostrar que es formalmente correcto. Esto significa que su funcionamiento es válido y es el esperado en cualquier ejecución posible, es decir, el resultado (output) guarda con los datos de entrada (input) la relación establecida por la especificación del problema [6, 9, 12, 13].

3.1 ALGORITMOS ITERATIVOS. PARADIGMA ITERATIVO

Aplicaremos este proceso, sin pérdida de generalidad, únicamente a los algoritmos iterativos cuyo esqueleto se corresponda con el esquema de la figura 1. En esta figura denotamos:

'Inic;' al grupo de instrucciones de inicialización del proceso.

'Cond' es la condición booleana que determina la permanencia en el proceso.

'SecInst' es la secuencia de instrucciones necesarias para la realización de una etapa en el proceso.

Muchos algoritmos pueden realizarse ajustándose a esta plantilla. No obstante la aplicación reiterada y local del proceso conduce de forma natural a la verificación de otros esquemas más complicados.

En el proceso de verificación [13] son precisos los siguientes elementos:

- Derivación de la condición invariante: A partir de la postcondición del programa buscaremos una relación invariante en cada ejecución del bucle entre los parámetros y las variables locales del algoritmo .

- Prueba inductiva: Consiste en demostrar que a partir de la precondición, la ejecución del algoritmo, conduce a la postcondición.

- Tamaño del problema: Consiste en demostrar que todos los bucles del algoritmo se ejecutan un número finito de veces, esto es, no se producen bucles infinitos y que por lo tanto el algoritmo para.

Figura 2: Esquema de verificacion en iterativos

Base de la inducción:
$\{PRE\}Inic\{INV\}$
Paso inductivo:
$\{INV \wedge Cond\}SecInst\{INV\}$
Consecución del objetivo:
$\{INV \wedge \neg Cond\} \rightarrow \{POST\}$
Definición de una función tamaño: $f \geq 0$
Decrecimiento estricto de f:
$f_k > f_{k+nv} : nv = nv(k) \in nat \forall k \in nat$

La figura 2 es una descripción de las pruebas a realizar para la consecución de la verificación de un bucle.

4 DISEÑO Y ESPECIFICACIÓN DE UN ALGORITMO DE CLASIFICACIÓN-DECISIÓN

El problema es el de decidir cuando la clasificación difusa de una imagen de píxeles es correcta, al menos en grado suficiente, de acuerdo con la opinión de un experto. Una primera propuesta a la especificación del algoritmo asociado es:

proc DecideyClasifica (**in** *im*: Timagen;
out *imc*:Tlista **of** Timagen; **out** *k*: **int**);
{PRE: *im* cumple las características de imagen }
{POST: *imc* es una lista de matrices resultado de la clasificación suficiente de la imagen *im*. La variable *k* devuelve la longitud de la lista *imc*}

Por supuesto la especificación formal de este algoritmo además del uso del cálculo de predicados difusos, requiere establecer el significado de clasificación suficiente.

4.1 DISEÑO DE UN ALGORITMO DE CLASIFICACIÓN-DECISIÓN

Las pautas para la verificación formal de un algoritmo difuso son las mismas que en el caso nítido teniendo en cuenta que en el contexto difuso el uso de los operadores habituales debe ser sustituido por las valoraciones difusas de los mismos. Por lo tanto el esquema de verificación de un algoritmo difuso coincide esencialmente con el de un algoritmo nítido.

Definimos un algoritmo difuso como aquél cuya especificación contiene algún predicado vago. Habitualmente encontraremos este tipo de predicados en la parte de precondicionamiento o postcondicionamiento del problema.

El siguiente algoritmo utiliza la opinión de dos expertos independientes para realizar la valoración del proceso. El algoritmo recibe una imagen y los niveles de confianza exigibles frente a la valoración de los expertos. El algoritmo devuelve una lista de matrices cuyos elementos son valores en el intervalo [0,1] y que representa la clasificación de la imagen así como un valor k donde se almacena la longitud de dicha lista, o lo que es lo mismo, el número de clases resultante.

El esquema de nuestro primer algoritmo representante de la primera parte del proceso es el siguiente: En primer lugar realizamos una llamada al proceso $ClasNit$ que produce una primera clasificación nítida de la imagen y le preguntaremos a un experto –el experto interno al proceso– su opinión sobre dicha clasificación.

La consulta al experto se realiza mediante el algoritmo *Decide* que devuelve una variable booleana *vale* a ***Cierto*** si el experto opina que la clasificación recibida es válida, lo que el experto decide tras comparar su propia valoración del algoritmo (α_{res}) con el valor propuesto (α_{int}) por el cliente. En ese caso, se entiende que el experto está de acuerdo con la clasificación realizada, por lo que acepta que el número de clases obtenido en la clasificación anterior sea correcto. En caso contrario, el experto devuelve *vale* a ***Falso*** y propone un número de clases para provocar una nueva clasificación con este parámetro prefijado. Por ello decidimos aumentar el número de parámetros de entrada del proceso añadiendo α_{int}.

Mientras el proceso no alcance la satisfacción del experto interno, esto es, mientras la variable *vale* sea ***Falso*** se realizarán clasificaciones difusas que el experto valorará.

proc $DecideyClasifica$ (**in** im: Timagen;
in $\alpha_{int} : [0,1]$;
out imc:Tlista **of** Timagen;
out k, k_0: **int**;
out α_{res}: [0,1]);
{PRE $_{DecideyClasifica}$}
ClasNit(im,imc,k);
$k_0 := k$;
Decide(im, imc, k, nk, vale,α_{res}*)*;
while¬$vale$ **do**
$ClasDif(im, nk, imc, k)$;
$Decide(im, imc, k, \alpha_{int}, nk, vale, \alpha_{res})$;
endwhile
{POST $_{DecideyClasifica}$}
endproc;

El algoritmo padre, esquema de todo el proceso desarrollado es el siguiente:

proc *ProcesoCompleto* (**in** im: Timagen;
in $\alpha_{ext}, \alpha_{int} : [0,1]$;
out imc:Tlista **of** Timagen;
out $\alpha_{res}, \alpha'_{res}$: [0,1];
out k_0, k: **int**);
{PRE $_{ProcesoCompleto}$}
DecideyClasifica(im, α_{int}*,imc,k,*α_{res}*)*;
Decide2(im, imc, k,α_{res}*, nk, vale,*α'_{res}*)*;
{POST$_{ProcesoCompleto}$}
endproc;

Este procedimiento realiza una llamada al proceso anterior de clasificación-decisión y a continuación consulta al experto externo su opinión sobre el resultado mediante una llamada al prodedimiento *Decide*2 de características similares a *Decide*. *Decide*2 devuelve *vale* a ***Cierto*** si el experto externo califica como válida la clasificación y además devuelve en el parámetro α'_{res} el valor que decide asignar a dicha clasificación.

Los valores α_{res} y α'_{res} son booleanos en el contexto difuso, por lo que su rango de posibilidades no sólo serán ***Cierto*** o ***Falso*** sino que será un rango del intervalo real [0,1].

A este tipo de rango deberán pertenecer también las variables α_{int} y α'_{ext}. Para generalizar todos los casos supondremos que estas variables toman valores en el intervalo [0,1] donde establecemos la imagen de las siguientes funciones de valoración que definimos a continuación para distinguir la actuación de los expertos:

◇ $Valor_{int}(imc, im)$ = valor que el experto interno asigna a la clasificación imc de la imagen im (= α_{res}), número de clases propuesto para la siguiente clasificación

◇ $Valor_{ext}(imc, im)$ = valor que el auditor, –experto externo–, asigna a la clasificación imc de la imagen im(= α'_{res})

◇ $Valor_{final}(imc, im, k_0, \alpha_{res}, \alpha'_{res})$ = función esti-

madora que produce una valoración de la clasificación teniendo en cuenta además el valor producido por la clasificación nítida: k_0, la última valoración realizada por el experto interno: α_{res} y la valoración realizada por el experto externo: α'_{res}

4.2 ESPECIFICACIÓN FORMAL DE *DecideyClasifica*

La precondición de este procedimiento deberá exigir que la imagen de entrada *im* cumpla las características de una imagen de píxeles. Para simplificar usaremos el predicado lógico $EsImagen(im)$.

La postcondición de *DecideyClasifica* debe indicar que el parámetro de salida *imc* contiene una clasificación difusa con valoración suficiente. De esta forma la postcondición de *DecideyClasifica* resulta:

$$\{POST : Valor_{int}(imc, im) \succeq \alpha_{int} \wedge k = long(imc)\}$$

donde gracias a la condición de generalidad de los predicados difusos sobre los nítidos podemos observar este predicado como postcondición válida cuando no se ejecuta el bucle –clasificación nítida– como cuando se ejecuta el bucle –clasificación difusa– .

4.3 ESPECIFICACIÓN FORMAL DE *ProcesoCompleto*

La precondición del proceso completo coincide básicamente con la precondición de *DecideyClasifica*, basta con añadir una cláusula más para el parámetro α_{ext} obteniéndose:

$$\{PRE : EsImagen(im) \wedge (0 \leq \alpha_{int}, \alpha_{ext} \leq 1)\}$$

En cuanto a la postcondición de *ProcesoCompleto* podemos establecer la siguiente:

$$\{POST : (\alpha_{res} \succeq \alpha_{int}) \wedge_d (\alpha'_{res} \succeq \alpha_{ext})$$

$$\wedge_d (f(im, imc, k_0, k, \alpha_{res}, \alpha'_{res}) \succeq g(\alpha_{int}, \alpha_{ext})\}$$

donde la primera cláusula indica que el experto interno admite como suficiente la clasificación de la imagen respecto al valor α_{int} prefijado por el cliente. De forma análoga la segunda cláusula indica que el experto externo admite como suficiente la clasificación de la imagen respecto al valor α_{ext} prefijado de antemano también por el cliente. La tercera cláusula comprueba una preferencia entre la valoración del proceso completo calculada mediante una función f que tendrá en cuenta los datos más relevantes indicados en sus argumentos y una función g de los niveles de confianza α_{int} y α_{ext}. Estas funciónes f y g pueden definirse como se desee en cada caso concreto y su comparación determinará junto con la primera cláusula un predicado difuso que nos dará información sobre la bondad del proceso completo de clasificación-decisión llevado a cabo.
La conjunción difusa de estas tres cláusulas conformarán el grado de adecuación del resultado del proceso a nuestro objetivo.

4.3.1 Especificación de los subprogramas

♣ Proceso de clasificación nítida
El procedimiento *ClasNit* tiene la siguiente especificación formal asociada:

proc *ClasNit* (**in** *im*: Timagen;
out *imc*:Tlista **of** Timagen; **out** *k*: **int**);
$\{PRE_{ClasNit} : EsImagen(im)\}$
$\{POST_{ClasNit} : EsCN(imc, im) \wedge (long(imc) = k) \wedge k_0 = k\}$

donde los predicados lógicos $EsImagen$ y $EsCN$ se pueden definir en lenguaje natural como:
$EsImagen(im) \equiv im$ cumple las condiciones de imagen digital
$EsCN(imc, im) \equiv imc$ es una clasificación nítida de los píxeles de la imagen *im*.
Este predicado se define formalmente por:

$$EsCN(imc, im) \equiv$$

$$\forall Mat \in imc[\forall i \in \{1..N\} \forall j \in \{1..M\}$$

$$(Mat[i, j] = 0 \vee Mat[i, j] = 1)] \wedge$$

$$\forall i \in \{1..N\} \forall j \in \{1..M\} (\sum_{Mat \in imc} Mat[i, j] = 1)$$

que corresponde con la definición clásica de partición.

♣ Proceso de clasificación difusa
De forma muy parecida podemos especificar el procedimiento de clasificación difusa:

proc *ClasDif* (**in** *im*: Timagen;
in *k*: **int**;
in/out *imc*:Tlista **of** Timagen;
out *nk*: **int**);
$\{PRE_{ClasDif} : EsImagen(im) \wedge (EsCN(imc, im) \vee EsCD(imc, im)) \wedge k = long(imc) \wedge imc = IMC\}$
$\{POST_{ClasDif} : EsCD(imc, im) \wedge (long(imc) = nk)\}$

donde se aparece un nuevo predicado $EsCD$ definido informalmente por

$EsCD(imc, im) \equiv imc$ es una clasificación difusa de los píxeles de la imagen *im*.
Y formalmente:

$$EsCD(imc, im) \equiv \forall p \in im, \sum_{C \in imc} \mu_C(p) = 1$$

donde $p = (i, j)$ representa un píxel de la imagen *im* y $\mu_C(p) = C[i, j]$ es el valor de pertenencia del píxel p a la clase C que será una matriz de la clasificación *imc*

♣ Consultas a los expertos

En cuanto a los procedimientos de consulta a los expertos *Decide* y *Decide*2 su especificación formal es la siguiente:

proc *Decide* (**in** *im*: Timagen;
in *imc*:Tlista **of** Timagen;
in k: **int**;
in α_{int}: [0,1];
out *nk*: **int**;
out *vale*: **bool**);
out α_{res}:[0,1];
$\{PRE_{Decide} : EsImagen(im) \wedge (EsCN(imc, im) \vee EsCD(imc, im)) \wedge k = long(imc)\}$
$\{POST_{Decide} : \alpha_{res} = Valor_{int}(imc, im) \wedge vale \Leftrightarrow (nk = k \wedge \alpha_{res} \succeq \alpha_{int})\}$

proc *Decide*2 (**in** *im*: Timagen;
in *imc*:Tlista **of** Timagen;
in k: **int**;
in α_{ext}: [0,1];
out *vale*: **bool**;
out α'_{res}:[0,1];);
$\{PRE_{Decide2} : EsImagen(im) \wedge (EsCN(imc, im) \vee EsCD(imc, im)) \wedge k = long(imc)\}$
$\{POST_{Decide2} : \alpha'_{res} = Valor_{ext}(imc, im) \wedge vale \Leftrightarrow (nk = k \wedge \alpha'_{res} \succeq \alpha_{ext})\}$

La verificación formal introducirá el cálculo de predicados difuso para evaluar el factor de certeza de cada aserto.

5 Verificación del algoritmo de decisión-clasificación

Uniendo todas las partes estudiadas podemos unir la clasificación difusa con la participación del experto externo en el algoritmo que contiene el proceso completo y que se muestra en el siguiente pseudocódigo:
proc *ProcesoCompleto* (**in** *im*: Timagen;
in $\alpha_{int}, \alpha_{ext}$:[0,1];
out *imc*:Tlista **of** Timagen;
out $\alpha_{res}, \alpha'_{res}$: **[0,1]**;
out k_0, k: **int**);
$\{\text{PRE}_{ProcesoCompleto}\} \Rightarrow \{\text{PRE}_{ClasNit}\}$
ClasNit(im,imc,k);
$k_0 := k$;
$\{POST_{ClasNit}\} \Rightarrow \{\text{PRE}_{Decide}\}$
Decide(im, imc, k, α_{int}*, nk, vale,* α_{res}*)*;
$\{\text{POST}_{Decide}\} \Rightarrow \{\text{INV: ... }\}$
while $\neg vale$ **do**
$\{\text{INV} \wedge \neg vale\} \Rightarrow \{\text{PRE}_{ClasDif}\}$
$ClasDif(im, nk, imc, k)$
$\{\text{POST }_{ClasDif}\} \Rightarrow \{\text{wp(INV)}\} \Rightarrow \{\text{PRE}_{Decide}\}$
$Decide(im, imc, nk, \alpha_{int}, k, vale, \alpha_{res})$
$\{\text{POST}_{Decide}\} \Rightarrow \{\text{INV: ...}\}$
endwhile
$\{\text{POST}_{DecideyClasifica}\} \Rightarrow \{\text{PRE}_{Decide2}\}$
Decide2(im, imc, k, $\alpha_{ext}, vale, \alpha_{res}$*)*;
$\{\text{POST}_{Decide2}\} \Rightarrow \{\text{POST }_{ProcesoCompleto}\}$
endproc;

el primer paso a realizar es la derivación de un invariante adecuado a la corrección del proceso:

$\{\text{INV: } (\alpha_{res}, nk) = Valor_{int}(imc, im) \wedge_d vale \Leftrightarrow (\alpha_{res} \succeq \alpha_{int}) \wedge_d vale \Leftrightarrow (k = nk)\}$

El invariante {INV} se cumple trivialmente tras las instrucciones de inicialización del algoritmo (la clasificación nítida y la primera llamada al experto interno). Para comprobarlo basta con comparar la postcondición del proceso *Decide* tras su primera ejecución:
$\{\text{POST}_{Decide}\} \Rightarrow \{\text{INV: ... }\}$
lo que constituye la base de la inducción que fundamenta nuestro proceso de corrección parcial.

En cuanto al paso inductivo, supondremos como hipótesis de nuestra inducción que se cumplen las condiciones:

$$\{INV \wedge \neg vale\}$$

por lo que se produce una entrada en el bucle y como consecuencia la ejecución de una nueva clasificación (difusa) para la imagen y una nueva consulta al experto interno. Tras la reclasificación puede haberse perdido la invarianza del predicado $\{INV\}$ pero la llamada al proceso *Decide* de consulta al experto restablecerá el valor de las variables de forma que se recupere la invarianza de nuevo.

En cuanto a la consecución del objetivo del bucle, es trivial observar que:

$$\{INV \wedge vale\} \Rightarrow POST_{DecideyClasifica}$$

con lo que se obtiene la corrección parcial completa de la primera parte del proceso, quedando únicamente pendiente la corrección de la secuencia de código relativa a la consulta al experto externo, esto es:

$\{\text{POST}_{DecideyClasifica}\} \Rightarrow \{\text{PRE}_{Decide2}\}$
Decide2(im, imc, k, $\alpha_{ext}, \alpha_{res}$*, vale)*;
$\{\text{POST}_{Decide2}\} \Rightarrow \{\text{POST }_{ProcesoCompleto}\}$

La postcondición del proceso completo, está constituida por tres cláusulas difusas que a su vez se conectan mediante el operador binario difuso $f_\wedge \equiv \wedge_d$ generalizado expuesto en la sección 3.2. de este artículo. De esta forma la postcondición del proceso completo se cumplirá en cierto grado y dicho grado a su vez puede servir como valoración final del proceso de clasificación-decisión desarrollado a lo largo de nuestro

artículo:

$$Valor_{final}(imc, im, k_0, \alpha_{res}, \alpha'_{res}) =$$
$$(\alpha_{res} \succeq \alpha_{int}) \wedge_d (\alpha'_{res} \succeq \alpha_{ext})$$
$$\wedge_d (f(im, imc, k_0, k, \alpha_{res}, \alpha'_{res}) \succeq g(\alpha_{int}, \alpha_{ext})\}$$

6 Conclusiones

Los algoritmos de clasificación-decisión difusos pueden ser debidamente diseñados, justificados y valorados mediante un método formal que detecte problemas y su posible solución generando diseños óptimos para procesos concretos. De esta forma podremos medir la bondad del algoritmo y establecer así un orden en el conjunto de algoritmos difusos asociados a un problema.

Referencias

[1] A. Amo, J. Montero y G. Biging: *Relevancia y redundancia en sistemas de clasificación difusa*. En: Proceedings Congreso Español sobre Tecnologías y Lógica Fuzzy, ESTYLF 2000 (A. Ollero, S. Sánchez, B. Arrue and I. Baturone, eds., Universidad de Sevilla, Sevilla); pp. 521-525 (2000).

[2] A. Amo, D. Gómez, J. Montero and G. Biging: Relevance and redundancy in fuzzy classification systems. Mathware and Soft Computing 8, 203-216 (2001)

[3] A.Amo, D. Gómez, J. Montero: Spectral fuzzy classification: a supervised approach. Mathware and Soft Computing 10:141-154 (2003)

[4] A.Amo, J. Montero, G. Biging y V. Cutello: Fuzzy classification systems. EUR. JOUR. OPER. RES. (to appear)

[5] A.Amo, J. Montero y V. Cutello: *On the principles of fuzzy classification*. En: Proceedings North American Fuzzy Information Processing Society Conference; 675–679 (1999).

[6] H.K. Berg y otros. Formal Methods of Program Verification and Specification. *Ed. Prentice Hall*, New Jersey, 1982.

[7] M. Broy y B. Krieg. Derivation Of Invariant Assertions During Program Development by Transformation. *ACM Transactions on Programming Languages and Systems*, Vol. 2, N 3, 321–327 1980.

[8] D. Gómez. *Algunas aportaciones sobre relaciones de preferencia*. Tesis doctoral, Universidad Complutense de Madrid, 2003.

[9] D. Gries. An Introduction to Current Ideas on the Derivation of Correctness Proofs and Correct Programs. *IEEE Software Engineering*, Vol. 2, N 4, 1976.

[10] J. R. Jensen: *Introductory digital image processing*. 1986 Prentice-Hall.

[11] A. Kaldewaij *Programming: The Derivation of Algorithms*, Ed.:Prentice Hall International, 1990.

[12] Z. Manna. The Correctness of Programs. *Journal of Computer and System Sciences*, Vol. 3, N 2, 1969.

[13] R. Peña Marí, *Diseño de Programas, Formalismo y Abstracción*, Ed.: Prentice Hall, 1999