TEMA 1

PERCEPCIÓN y CONTROL a NIVEL de CONOCIMIENTO

J. Mira

1.1. Introducción y Contexto

El propósito de este material didáctico es ayudar al alumno a comprender los contenidos de la signatura, guiándole también en el uso de material complementario. Incluiremos también conocimiento de naturaleza tutorial, de acuerdo con la metodología propia de la enseñanza a distancia. En la enseñanza presencial, el alumno dispone, en general, de la posibilidad de interacción en tiempo real con el profesor para preguntarle las cuestiones que no entiende. Aquí, tenemos que imaginar cuáles pueden ser esas cuestiones e intentar contestarlas, haciendo las exposiciones más redundantes, justificando la razón de cada módulo de material, viendo qué conocimientos previos necesita el alumno, dando pistas para la autoevaluación, etc…

La asignatura se enmarca en la línea de especialización en Inteligencia Artificial (IA), donde el alumno ha tenido que estudiar primero Lógica, Teoría de Autómatas e Introducción a la IA, junto a otras asignaturas. Además, es razonable suponer que también haya estudiado Sistemas Basados en Conocimiento I (optativa del primer cuatrimestre del tercer curso) y que pueda estar compartiendo esta asignatura con Sistemas Basados en Conocimiento II.

Por consiguiente, podemos aceptar que los estudios previos del lector de este material docente incluyen: (1) los aspectos metodológicos básicos de la teoría de niveles y dominios de descripción de un cálculo, (2) los mecanismos de búsqueda en grafos y las técnicas de representación del conocimiento mediante reglas y marcos, (3) los aspectos básicos del desarrollo de sistemas basados en conocimiento (tareas, métodos y capas de conocimiento del dominio) y (4) ciertos rudimentos sobre redes de neuronas artificiales, en particular los modelos básicos del perceptrón multicapa y el aprendizaje supervisado por retropropagación del gradiente.

Sobre este panorama de fondo vamos a plantear ahora el propósito y el contenido de la asignatura de “Percepción y Control Basados en Conocimiento” (P&C). Como en toda tarea

computacional, las preguntas que debemos hacernos a la vez que ayudamos a nuestros alumnos a que sepan contestarlas, son las siguientes:

PERCEPCIÓN

P.1. ¿Cómo se describe la tarea de percepción en lenguaje natural (LN), a nivel de conocimiento (N. de C.) y en el dominio de un observador externo a la computación (OE)?.

P.2. ¿Qué aspectos de esta descripción son computables?. Es decir, ¿cuál es la versión máxima de un modelo conceptual del proceso de percepción del que, posteriormente, podamos obtener un programa?

P.3. ¿Cómo construimos ese modelo? (usando, por ejemplo, una versión simplificada de la metodología Common KADS, separando estructura de inferencias (E. de I.) de conocimiento del dominio (C. del D.), etc…

P.4. ¿Cómo reducimos ese modelo al nivel de los símbolos?[1].

   a. Por la vía "simbólica".

   b. Por la vía "conexionista".

   c. Usando métodos híbridos.

P.5. Finalmente, ¿Cómo conectamos con las aplicaciones reales en el mundo Físico?. (i.e. sensores, efectores, etc…).

las mismas preguntas se repiten ahora para la tarea de CONTROL.

CONTROL

C.1. ¿Cómo se describe la tarea de control en LN, a N. de C. y en el dominio del OE?.

C.2. ¿Qué aspectos de C1 son computables?.

C.3. ¿Cómo construimos ese modelo de pericia (E.I. del Método, C. del D.)?

C.4. ¿Cómo reducimos C.3 al nivel de los símbolos? (programas, redes neuronales, híbridos, …).

C.5. ¿Cómo los aplicamos?. (robots, sensores, efectores, etc…).

[1] De hecho, lo simbólico y lo conexionista son, simplemente, distintos tipos de métodos de solución de problemas, distintos paradigmas de programación que dependiendo de la naturaleza de un problema, del conocimiento del que se dispone al comienzo o de la etapa en la que se encuentra el proceso de solución, aconseja el uso de redes neuronales artificiales y/o métodos simbólicos tipo "establece y refina" o "propón-actúa-modifica", o establece mediante búsqueda en grafos y refina mediante el ajuste de parámetros usando una red neuronal y un algoritmo de ajuste por mínimos cuadrados y descenso del gradiente.

Si ya sabemos modelar y programar las tareas de percepción y las de control ¿cómo podemos ahora integrarlas en una aplicación real?. en nuestro caso elegimos la rebótica autónoma como campo de integración de las tareas de percepción y acción. La razón más fuerte es que es en el caso de los robots donde se integran de forma más natural y necesaria las tareas que tienen que ver con la representación de un medio externo, en general no estructurado, con el propósito de navegar y realizar acciones en el mismo.

De hecho, la robótica es el paradigma más completo de comportamiento artificial y el banco de pruebas más completo de la IA aplicada. La computación asociada a un robot debe ser capaz de modelar y programar (1) la representación sensorial de un medio, incluyendo la descripción física de un conjunto de sensores (sonar, infrarrojos, de impacto, …) de los que el más complejo y completo es el visual, (2) el preprocesado de bajo nivel de esta representación, (3) la interpretación de las escenas y el resto de los procesos perceptivos de alto nivel, (4) el procesado central de funciones de decisión y planificación espacial y temporal y (5) el control del proceso de navegación, de acuerdo con la planificación de la acción y de la constante realimentación sensorial en ese medio. (6) Finalmente, hay de nuevo otra etapa de bajo nivel asociada al control de secuencias temporales de órdenes para los motores, los brazos articulados y el resto de los efectores que conectan el modelo computacional con el mundo físico en toda aplicación.

Queremos resaltar aquí la enorme importancia de los sensores y los efectores y el resto del conocimiento necesario para hacer útil un desarrollo en P&C dentro de la IA. El sólo modelo computacional de los procesos de percepción, planificación y control ("el alma", el Sistema Nervioso) suficiente si en la etapa de implementación no se resuelven los problemas asociados a los sensores y efectores ("el cuerpo", los pies, las manos) que le permitan al robot interactuar de forma eficiente en un entorno físico real, en general complejo y no estructurado.

Planteado así el problema, no es difícil comprender el alcance potencial de esta asignatura de P&C basados en conocimiento. Sin embargo, el carácter de materia cuatrimestral y optativa nos obliga a limitarnos a las componentes básicas y fundamentales. Básicas, en el sentido de sencillas, elementales y previas a otras materias de nivel superior y fundamentales en el sentido de fundamentos. Es decir, de aquellos conocimientos que ayudan al alumno a situarse con claridad conceptual ante la naturaleza del problema del modelado computable de las tareas de P&C.

Guiados por estos criterios (lo básico y los fundamentos), dedicaremos el resto de este tema primero a recordar (en el apartado 2) el modelo general de computación en un nivel, (en el 3) la teoría de niveles y dominios de descripción de un cálculo, (en el 4) la estrategia de reducción de los modelos usando métodos simbólicos, (en 5) la reducción de los modelos por la vía conexionista, (en 6) la descomposición de la percepción en otras tareas más elementales usando distintos métodos y en el apartado 7 la descripción de la tarea de control (incluyendo la supervisión) usando métodos simbólicos o conexionistas.

Terminado este primer capítulo metodológico y de modelado conceptual a nivel de conocimiento, estudiaremos los sensores no visuales y después la visión de bajo nivel. La visión de alto nivel sólo se estudiará de firma cualitativa, a través de la introducción del modelo del medio. De hecho, la mitad del nombre de la asignatura (percepción) es algo pretenciosa. La percepción es una tarea de alto nivel y realmente compleja que exige la integración plurisensorial y la inyección de conocimiento externo para "interpretar" los

objetos, "comprender" la escena y asociarla con contenidos de memoria. Como sabemos que al final de estos modelos siempre nos vamos a encontrar con tablas , grafos o autómatas finitos, es evidente que el término percepción queda muy limitado en significado cuando pasa de la esfera de lo humano al campo de la computación y la robótica.

La tarea de percepción termina cuando somos capaces de construir una cierta representación del medio, incluyendo la propiocepción. Es decir, un robot debe tener una serie de ficheros en los que se carga la evolución temporal de las señales que captan sus sensores y de la posición de él mismo en ese medio. A esto le hemos llamado, desde los tiempos de Kenneth Craik, modelo del medio y computacionalmente puede representarse por un grafo.

Ahora empieza la tarea de planificación y control que utiliza los datos de la representación interna del medio. En su forma más elemental, controlar es dar una orden y garantizar que se cumple. Para ello se usa un esquema inferencial en forma de lazo de realimentación negativa que compara la señal de mando con una medida de la respuesta y actúa intentando disminuir la señal de error. Así, introduciremos primero los rudimentos de un sistema de control lineal, descrito en el dominio del tiempo y pasaremos después a su representación mediante reglas y redes de neuronas artificiales que nos permiten programar los modelos sin la exigencia analítica de conocer las transformadas de Fourier y Laplace y los modelos matemáticos de los procesos no lineales. esta es una exigencia docente básica, dado el perfil de conocimientos previos que se suponen al alumno que cursa esta asignatura. Es decir, no le vamos a exigir a nuestros alumnos que sepan plantear y resolver el conjunto de ecuaciones diferenciales asociadas al control de un brazo manipulador, ni que sepa calcular las funciones de transferencia de la dinámica de un robot. Sin embargo, sí que le vamos a exigir que sepa modelar y programar mediante reglas o redes neuronales y usando los entornos de desarrollo adecuados, un modelo elemental de planificación y un proceso de supervisión, por ejemplo. La razón es que para esta segunda tarea, sabemos que nuestros alumnos disponen de los conocimientos previos necesarios. Es decir, saben representar conocimiento usando reglas y marcos; saben programar usando entornos, saben modelar tareas, etc.

No nos gustaría acabar este apartado de introducción sin hacer una llamada de atención a nuestro alumnos sobre una actitud que ha permanecido desgraciadamente invariante durante todo el desarrollo de la inteligencia artificial. Nos referimos al uso constante de términos antropomorfos, tomados de la biología y la psicología para designar entidades del mundo de la electrónica y de la computación sin razones serias para hacerlo. Así, hablamos de redes de neuronas artificiales cuando estamos hablando de operadores analógicos (sumas y umbrales) que ni de lejos se acercan a las funcionalidades de las neuronas reales. Hablamos de inteligencia artificial cuando no está claro que sepamos en qué consiste la inteligencia natural y mucho menos que hayamos sido capaces de mimetizarla seriamente en un computador. Hablamos de vida artificial cuando apenas estamos arañando los secretos del código genético y, últimamente, no contentos con el saqueo a que hemos sometido a los términos propios de la esfera semántica de lo cognitivo, hemos entrado a saco en la esfera emocional. Así, hemos pasado de buscar el pensamiento artificial a buscar el sentimiento artificial, la máquina emocional e intencional, con agentes benévolos que tienen motivaciones, propósitos e intenciones en el medio externo, etc.

Esta componente de espectacularidad y misterio asociada a la *IA* ha traído beneficios a corto plazo pero grandes perjuicios a medio y largo plazo para la docencia y la

investigación, porque nos ha distraído quitándonos tiempo, esfuerzo y recursos para buscar en la verdadera dirección, menos espectacular que la usual, pero probablemente mucho más seria y rentable a largo plazo.

Y es una pena porque la realidad supera a la ficción. Es decir, porque el camino usual en la ciencia y en las ingenierías de la materia y la energía puede ofrecer resultados más espectaculares, sólo que con nombres más modestos. Así, llamando computación neuronal a la computación modular, de grano pequeño, distribuida y paramétrica con capacidad para ajustar esos parámetros mediante funciones de coste global, no se pierde nada de su eficacia. Análogamente, también seríamos más eficaces distinguiendo dos partes en la *IA* y definiéndolas usando sólo términos inequívocos:

IA.I Modelado computable de parcelas del conocimiento humano no analítico en tareas técnicas de mediana complejidad (clasificación, diagnostico, monitorización, planificación, control, supervisión, …). En nuestro caso, modelado computable de una representación del medio ("percepción") y del proceso de planificación y control de la trayectoria).

IA.II Simulación e implementación de sensores y efectores análogos a los que poseen los seres vivos, inspirándonos en la Biología o buscando su síntesis por procedimientos distintos.

Con esta concepción de la *IA*, que nos dice que su propósito es primero **modelar** y después **programar** esos modelos, de forma análoga a cómo se modela y programa la solución de ecuaciones diferenciales o cualquier otra aplicación, aunque con las complicaciones propias de usar un conocimiento en general no analítico, tendríamos trabajo serio suficiente para muchos años.

Otra característica negativa de todo el desarrollo de la *IA* ha sido la prisa y la consiguiente falta de metodología y teoría. Hemos querido construir aviones sin saber Física de primer curso de ingenieros aeronáuticos. Así, muchos laboratorios de *IA* merecerían el nombre de "laboratorio de resultados irreproducibles", distintos de los que se obtienen en otros laboratorios y por otros investigadores.

Para terminar esta introducción me gustaría recordar a nuestros alumnos la forma en la que deben estudiar esta asignatura:

I. Construyendo un esquema metodológico con distintos niveles y dominios de descripción que los permita "llevar la cuenta" de forma honesta de las distintas entidades y relaciones que manejan al modelar una tarea, de las tablas de semántica correspondientes y de la causalidad que opera a cada nivel. Así se eliminaría el uso de una misma palabra, (propósito, por ejemplo) con distinto significado en el contexto humano y en lenguaje natural ("propósito"), del significado cuando se introduce en un programa (**propósito**), donde queda reducido a un autómata finito o a un grafo.

II. Teniendo claro que la tarea fundamental en esta asignatura es **modelar conocimiento** sobre los procesos de percepción y control usando todas las técnicas matemáticas, lógicas y de programación que nos ofrece la ciencia de la computación.

III. Procurando eliminar los términos antropomorfos cuando los conceptos a los que hacen mención no sean biológicos o pertenecientes a las ciencias del comportamiento.

IV. Finalmente, recordando los métodos y el rigor de ciencias duras tales como la física e ingeniería de la materia para intentar conseguir algo análogo en la nueva "Ciencia e Ingeniería del Conocimiento" que es lo que creo que debería llegar a ser la ahora llamada *IA*.

Este énfasis nuestro en preguntarnos siempre por cual es el modelo que subyace a la computación y cual es la tabla de semántica que se queda fuera creemos que puede contribuir a aclarar de que estamos hablando cuando decimos que hemos mecanizado la tarea de percepción, la planificación o el proceso de control de efectores. Es decir, puede contribuir a distinguir las funcionalidades reales de un programa (sean las que fueren) del valor añadido por un autor "optimista" y "generoso" a la hora de valorar externamente ese programa. Es necesario saber discriminar entre *lo que calcula de verdad un programa* (el conocimiento que hemos sido capaces de modelar y representar de forma computable) y el *resto del* ***conocimiento*** que se quedó (*necesariamente*) en el nivel de conocimiento al codificar las entidades del modelo en términos de las variables y los operadores de un lenguaje formal y que se añade de nuevo al interpretar los resultados de la computación, haciendo ***emerger*** esos significados al usar, de forma inversa, las mismas tablas de semántica que se usan en la reducción.

1.2. El modelo General de Computación en un Nivel.

El modelo general de computación [1,2,3] afirma que**,** toda la fenomenología de un nivel se puede describir mediante la interacción medio/sistema, tal como se muestra en la figura 1, donde el medio es a su vez otro sistema que puede ser descrito de la misma forma.

> De hecho, cada partición medio/sistema **define un compartimento de un nivel**, específica un conjunto de señales del medio (que entiende el sistema) y especifica también el conjunto de respuestas del sistema (que entiende el medio).

Es decir, cada partición medio-sistema dentro de un nivel queda caracterizada por un lenguaje formal común con el que se describe la interacción medio-sistema (la dinámica de las señales que intercambian).

Por convenio, llamamos ***medio*** a lo que estimula y ***sistema*** a lo que responde y el comportamiento del sistema se describe en términos de un conjunto de ***variables de entrada*** $X = \{x_i(t)\}$ que deben ser medibles, un conjunto de ***variables de salida*****,** $Y = \{y_j(t)\}$ que también deben ser medibles, y un conjunto de ***reglas de transformación***:

$$R = \left\{ f_{ij}^{k}(t), g_{ij}^{k}(t) \right\}$$

que, de forma inequívoca, realizan procesos de cálculo de naturaleza analítica y/o lógico-relacional, sobre las variables de entrada y los contenidos de memoria, $M = \{m_n(t)\}$, para generar los valores de las variables de salida:

$$y_j(t + \Delta t) = f_{ij}^{k}\left[x_i(t), m_k(t) \right]$$

y modificar los contenidos de la memoria,

$$m_k(t+\Delta t) = g_{ij}^k\left[x_i(t), m_k(t)\right]$$

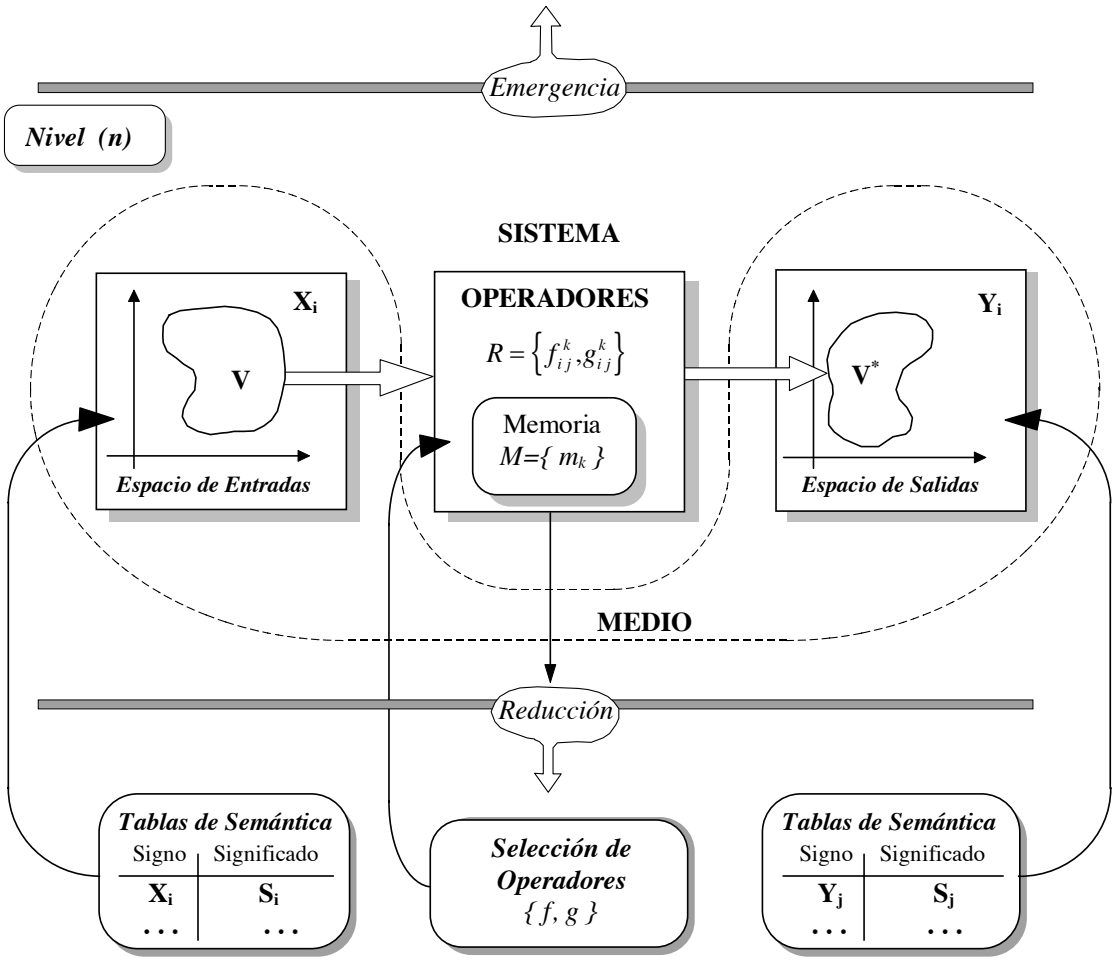


Fig. 1. Modelo computacional en un nivel. Los espacios de entrada y salida son en general, espacios de representación con tablas de semántica dependientes del nivel y del conocimiento que se quiere modelar. los operadores deben ser acordes con la naturaleza de los datos experimentales.

Todo modelo computable en un nivel puede entonces describirse en términos de un conjunto de ***señales*** (variables x_i, y_j y m_k) que representan la información (los datos) y un conjunto de reglas (operadores f_{ij}^k y g_{ij}^k) que especifican de forma **"*clara*, *precisa*, *completa* e *inequívoca*"** los procesos analíticos o lógicos-relacionales que se usan para transformar cualquier secuencia de representaciones de entrada $\{x_i(t), x_i(t+\Delta t), x_i(t+2\Delta t), \ldots\}$ en la correspondiente secuencia de representaciones de salida $\{y_j(t), y_j(t+\Delta t), y_j(t+2\Delta t), \ldots\}$, sin necesidad de ninguna conexión causal con el ***significado*** de las variables. Es decir, en principio las variables $\{x_i\}$ e $\{y_j\}$ podrían interpretarse como ***magnitudes físicas*** que sirvieran de ***soporte material*** de la computación. Estaríamos entonces en un ***nivel físico***, en el hardware de la computación, en la electrónica digital y la arquitectura de computadores donde las variables $x_i(t)$ e $y_j(t)$ son señales binarias con sólo dos valores posibles (o ó 5 voltios, por ejemplo), asociados a dos estados lógicos ("0" y "1").

También podríamos estar en el mundo de la electrónica analógica donde las señales $x_i(t)$ e $y_j(t)$ son continuas en un determinado intervalo y las funciones f y g representan -por ejemplo- los procesos de ***comparación***, ***amplificación***, y ***medida*** de un circuito de control. En este caso, la computación se modela en el dominio del tiempo mediante integrales de convolución y en el dominio de las frecuencias mediante "funciones de transferencia". Ya hemos mencionado en la introducción que aquí no haremos énfasis en el modelado analítico, pero lo citamos ahora para que el alumno comprenda la diferencia entre el modelado conceptual de un proceso computable y la fase posterior de selección de operadores , que nos lleva a un modelo físico (una implementación) de esa computación.

Obsérvese sin embargo que no hay nada en el modelo que nos obligue a esta interpretación. Es decir, las reglas f_{ij}^k y g_{ij}^k que enlazan los espacios de representación son independientes de la semántica de otros niveles y para conseguir sus resultados formales no necesitan hacer referencia alguna a los significados de las variables. Esto implica que, si en vez de hablar de variables físicas de entrada y salida, hablamos de ***espacios de representación*** de las entradas y las salidas, podríamos estar hablando de un modelo de computación al nivel de los símbolos. Es decir, de un programa. En este caso que, además, es el usual, todo modelo de percepción o control termina siendo un programa. Un conjunto de conversores analógicos-digitales (AD) han transformado los valores continuos de las variables (x,y) en una secuencia de palabras digitales que el programa transforma en otra secuencia digital. Finalmente, antes de volver al mundo físico, un conversor digital-analógico recupera el carácter analógico (continuo) de las señales que van a actuar sobre los efectores.

Si subimos más arriba, el modelo general de la figura 1 puede describirse en **lenguaje natural** y a nivel de conocimiento, suponiendo que se lo estamos contando a otro humano que conoce la materia, junto con las herramientas formales propias de la lógica y las matemáticas. Por ejemplo, como estamos haciendo nosotros ahora que escribimos un material didáctico para que nos comprendan nuestros alumnos que poseen un "compilador de lenguaje natural" y entienden el castellano. La clave está en las ***tablas de semántica*** que usemos para describir el ***significado*** de esas variables y, consecuentemente, el de los **procesos** representados por los operadores que las enlazan. En cada nivel las variables $\{x_i\}$ e $\{y_j\}$ tienen un ***significado*** diferente y los operadores trabajan de acuerdo con una ***causalidad*** también característica del nivel.

El procedimiento seguido en este apartado es el propio de un proceso de análisis. Sin embargo, lo usual en computación es seguir el camino inverso (síntesis). Es decir, partimos de un conjunto de especificaciones funcionales en lenguaje natural y buscamos su reducción a un programa. Como la máquina sólo dispone de compiladores para lenguaje de programación, nos vemos obligados a ***precisar*** mucho los modelos y a ***limitar*** la semántica de las variables.

1.3. Niveles y Dominios de Descripción en Computación

1.3.1. Niveles

Ahora es usualmente aceptado en el campo de la *IA,* tal como vimos en la asignatura *Introducción a la IA* [4] que para sintetizar un modelo computable es necesario distinguir e integrar al menos, tres niveles de descripción tal como se ilustra en la figura 2:

I) Una teoría de cálculo (Nivel de ***Conocimiento*** en Newell).

II) Un algoritmo (Nivel de los ***Símbolos*** –programa– en Newell).

III) Una implementación electrónica (Nivel ***Físico*** en Newell)

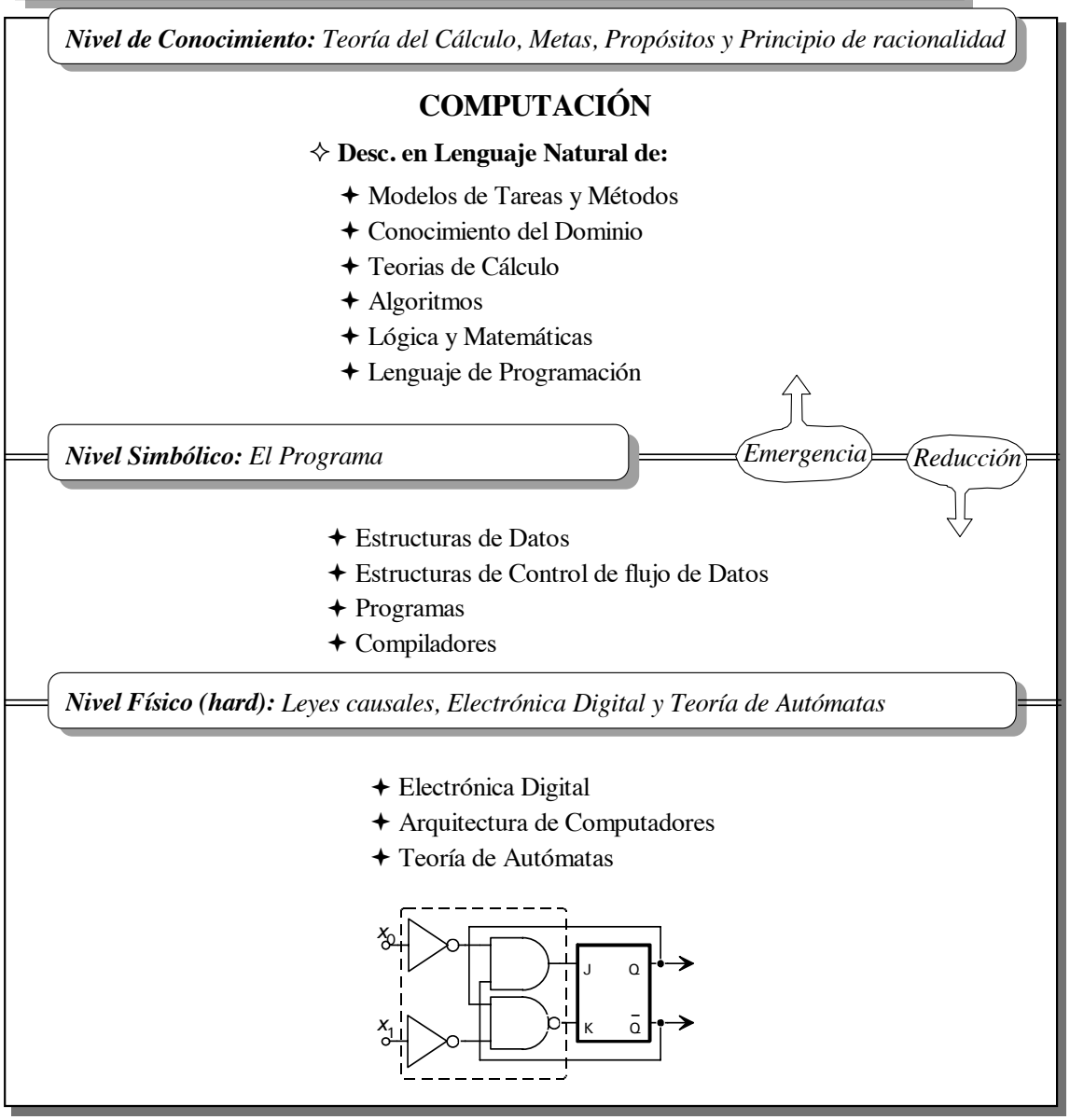


Fig. 2. Niveles de descripción de un cálculo. I. Nivel de Conocimiento. II. Nivel de los símbolos computables (lenguaje de programación). III. Nivel físico.

En el ***primer nivel*** tenemos los fundamentos teóricos del modelo que queremos hacer computable descritos en lenguaje natural y un posible esquema conceptual del proceso, junto con los conceptos propios del dominio. Para conseguir comprender un determinado proceso debemos empezar describiendo su propósito, las entidades y sus relaciones y algunas pistas sobre los principios en los que puede apoyarse. Este primer nivel de David Marr [5] se engloba en el ***nivel de conocimiento*** propuesto por Newell en 1981 [6]. Nada

podrá ser modelado computacionalemte si previamente no ha sido descrito de forma "clara, completa, precisa e inequívoca" en lenguaje natural y a nivel de conocimiento.

El ***segundo nivel*** de Marr (***representación*** y ***algoritmo***) incluye la descripción algorítmica del modelo anterior y se corresponde, aproximadamente, con el ***nivel de los símbolos*** propuesto por Newell (el ***programa***). La diferencia entre las propuestas de Marr y Newell es básica a nivel conceptual ya que lo que nos dice Newell es que prácticamente toda la parte relevante de una computación se queda en el nivel de conocimiento, incluyendo el algoritmo. Así, la propuesta de Newell es clara: "el nivel de la símbolos es el programa", todo lo demás queda "por encima" (el conocimiento) o "por debajo", (la máquina electrónica o la red neuronal).

El ***tercer nivel*** tiene que ver con todo el proceso de implementación que nos lleva del programa a los procesadores físicos. Su contenido corresponde a las asignaturas de Electrónica Digital y Arquitectura y Tecnología de Computadores y no nos extenderemos más aquí. En aplicaciones de Percepción y Control en ***tiempo real*** puede ser necesario bajar a este nivel para optimizar una implementación, pero no en el caso de las aplicaciones que se consideran en esta asignatura.

1.3.2. Dominios

Queremos saber qué estamos diciendo de hecho cuando decimos que un modelo de visión artificial es computable o cuando decimos que un programa de *IA* "diagnostica" como un cardiólogo o cuando decimos que un robot "percibe" su medio, "controla" su navegación y tiene "propósitos" en ese medio. Para ello vamos a completar la descripción en tres niveles de todo modelo computable recordando la figura del ***observador*** externo a la computación que superpone dos sistemas de referencia, dos dominios, sobre los tres niveles (tal como fue estudiada en el segundo tema de la asignatura de Introducción a la IA). Estos dos dominios de descripción son:

I) El dominio propio de cada nivel (*DP*)

II) El dominio del observador externo (*DOE*) que primero codifica y después interpreta el significado de la computación.

Tenemos así un edificio epistemológico de tres plantas (***conocimiento***, ***símbolos*** y ***física***) y dos pisos por planta (*DOE* y *DP*) de forma que al modelar, programar e interpretar no debemos confundirnos ni de planta ni de piso. De lo contrario, los resultados del modelo o del programa de *IA* serán confusos y, probablemente, erróneos.

La introducción de la figura del observador y la distinción entre una fenomenología y su descripción, procede de la física y ha sido reintroducida y elaborada en el campo de la biología por Maturana [7] y Varela [8] y en la *IA* y la computación neuronal por Mira y Delgado [9,10,11].

En el dominio propio (*DP*), que se ilustra en la columna derecha del edificio de la figura 3, todo lo que ocurre en los distintos niveles es causal y no arbitrario. Las relaciones espacio-temporales entre los valores de las distintas variables son relaciones de ***necesidad***. No pueden ser otras que las que su estructura determina. La semántica, además es propia e inherente al nivel. Estructura y función coinciden y ocurre "lo que tiene que ocurrir".

El ***DP del nivel físico*** es quizás el más evidente. En Electrónica, los inversores invierten y los contadores cuentan, porque el circuito está construido así. Las leyes del comportamiento observable de estos circuitos y su casualidad son las de la lógica y la teoría de autómatas finitos. No bajamos a la electrónica física ni a la física de los dispositivos (transistores) con los que se han construido esos inversores.

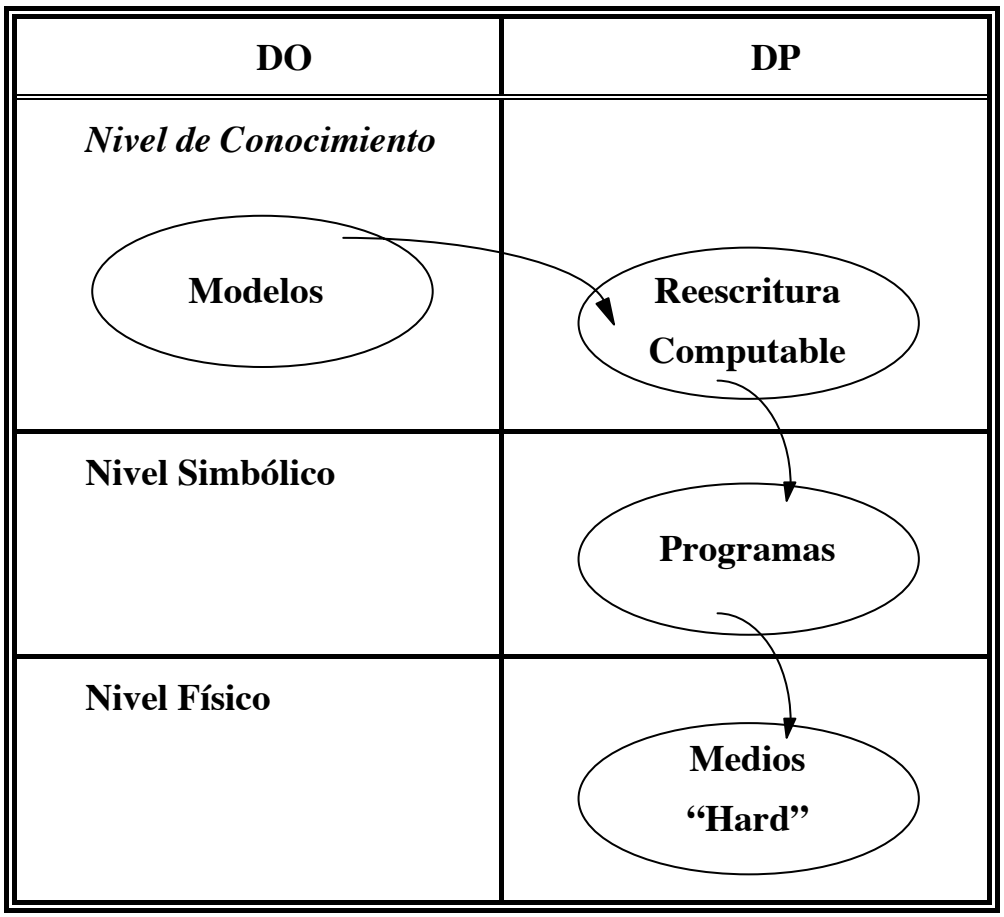


Fig. 3. Superposición de los dos dominios de descripción de un cálculo (el propio y el del observador externo) a los tres niveles que hemos introducido previamente (conocimiento, símbolos y electrónico).

Si subimos ahora del nivel físico al nivel de los símbolos, vuelve a repetirse el proceso. El nivel de los símbolos en computación lo constituye el programa y ningún programa puede salirse de la sintaxis, la semántica y la pragmática del lenguaje de programación con el que ha sido escrito porque de lo contrario no sería aceptado por su compilador y, por consiguiente, no podría pasar al nivel físico. No podría ejecutarse. Lo que en el nivel físico eran niveles de tensión en circuitos digitales, ahora son valores de verdad (1 = verdadero, 0 = falso) en expresiones lógicas. Y esa es la única semántica del nivel.

Cuando hablamos de computación, el *DP* contiene a los niveles físico y simbólico. No creemos en la existencia de un nivel de conocimiento en el *DP* de un computador (su "mente").

Veamos ahora las descripciones correspondientes en el dominio del observador externo. Su existencia es evidente ya que estamos hablando simplemente de la **persona que hace el modelo** y lo programa después. Las descripciones de una computación siempre se desarrollan en el ***dominio del observador*** (*DO*), ***en lenguaje natural y a nivel de conocimiento*** usando la semántica propia del lenguaje natural y del conocimiento que tenemos sobre las entidades del dominio. Es decir, la primera versión en el proceso de síntesis de un modelo computable siempre se escribe en lenguaje natural, haciendo uso de las herramientas formales de las que disponemos (lógica y matemáticas) y con la semántica propia de lo humano. Cuando hablamos de presión, volumen o temperatura y escribimos PV=nRT, usamos una fórmula pero damos por supuesto, además, que P, V y T tienen el significado usual en Física. Si además estamos usando estas variables en un dominio médico damos por supuesto también el significado de "presión intracraneal", "volumen pulmonar" y "temperatura alta" (fiebre), porque esas variables tienen una nueva esfera semántica superpuesta a la que usaría un físico, por ejemplo.

Las descripciones de una computación en el ***dominio del observador*** (*DO*) siempre usan la ***semántica*** y la ***causalidad*** del lenguaje natural, incluyendo la lógica y las matemáticas y los metalenguajes del conocimiento específico de la tarea (i.e. visión artificial) y el dominio (i.e. imágenes de radiología pulmonar) necesarios para especificar el modelo de conocimiento del problema que se quiere hacer computacional ("un programa que diagnostique en patologías pulmonares", por ejemplo).

En computación en general y en *IA* en particular la teoría de ***niveles y dominios*** de descripción de un modelo computable encuentra su uso natural en procesos de ***ingeniería directa***, en los que partimos de un conjunto de especificaciones funcionales sobre la solución de un problema que queremos hacer computable, lo modelamos a nivel de conocimiento y construimos después un programa que, con el compilador adecuado, se ejecuta en una máquina física. Es decir, el mapa de niveles y dominios se recorre de izquierda a derecha y de arriba hacia abajo. Empezamos con un modelo en el *DO* y a nivel de conocimiento ("3º izquierda") y pasamos al *DP* a nivel simbólico ("2º derecha"). Después el compilador la baja al *DP* del nivel físico ("1º derecha"). Y esto es todo señores. Ni más ni menos. Aquí acaba la descripción general de la tarea fundamental en *IA* aplicada.

El resto son elucubraciones y funcionalidades añadidas sin justificación, usando inadecuadamente conceptos con significado propio de la esfera semántica humana. Y lo sorprendente del caso es que la realidad supera a la ficción. Es decir, que desbrozado el problema de la IA aplicada, la tarea clara, nítida y transparente que aparece ante nuestros ojos (modelar conocimiento) tiene todavía el atractivo y la complejidad real suficiente para atraer y apasionar a muchos alumnos, profesores e investigadores durante muchos años.

En esto consiste el viejo sueño griego de mecanizar el pensamiento, en bajar las escaleras desde el tercero izquierda al primero derecha, volver a subir cuando los resultados no son satisfactorios (validar, evaluar, remodelar), volver a bajar, volver a subir, …

1.4. La Reducción de los Modelos usando Métodos Simbólicos

El problema de la IA aplicada a cualquier dominio de conocimiento, por ejemplo a la visión artificial o a la robótica o al diagnóstico en Medicina, siempre es el mismo a nivel metodológico. Se trata de **resolver un problema** que, en general, no está bien planteado, su

solución no es fácil usando técnicas algorítmicas convencionales o la explosión combinatoria hace ineficientes otros métodos.

Por consiguiente, una parte importante del trabajo está asociada a la ***formulación*** adecuada del problema; es decir, a obtener algo parecido a un conjunto de ***especificaciones funcionales*** a partir de una descripción en lenguaje natural de lo que queremos que haga el sistema. No se nos olvide que la meta final es conseguir un sistema electrónico con un conjunto de sensores y/o efectores y un ordenador con un programa que "duplique" o "mimetice" de forma razonable la tarea humana de partida ("ver", "andar", manipular objetos, etc…).

Si nos centramos ahora en el ***programa***, el planteamiento del problema no es difícil. Queremos enseñar a nuestros alumnos a que, a partir de un conjunto de especificaciones funcionales sobre las tareas de **P&C**, nos generen un programa que las satisfaga, como en cualquier otra rama de la informática aplicada.

Lo peculiar de la IA aplicada es que los problemas que aborda no son fáciles de especificar y por eso se ha desarrollado toda una metodología más costosa que en otras ramas (tales como las bases de datos o el cálculo numérico). Esta metodología se centra en los siguientes puntos:

1. Énfasis en la descripción clara y precisa en lenguaje natural y a nivel de conocimiento, recordando que estamos en el "3º izquierda" (dominio propio) y que queremos llegar al "2º derecha" (programa en lenguaje de programación-dominio propio y nivel de los símbolos-).
2. Esfuerzo en encontrar "***componentes reutilizables***" en el diseño. Así, se empezó distinguiendo entre "***tareas genéricas***" y conocimiento específico del dominio. Por comodidad de lectura incluimos aquí la figura 4 (traída del texto base en Introducción a la IA) que resume la biblioteca de ***tareas genéricas*** en las que, en principio, deberemos empezar a buscar para resolver nuestros problemas de **P&C**. Así por ejemplo, el reconocimiento de objetos puede formularse a partir de tareas de análisis tales como el diagnóstico o la clasificación.
3. ***Reconocimiento*** del excesivo "***tamaño***" de las tareas genéricas, y búsqueda de distintos ***métodos*** ("Problem Solving Methods", PSM's) para descomponer esas tareas en otras sub-tareas más elementales. Por ejemplo, diremos más adelante que la tarea de ***reconocimiento*** de objetos puede descomponerse en "preproceso-segmentación-métrica y decisión".

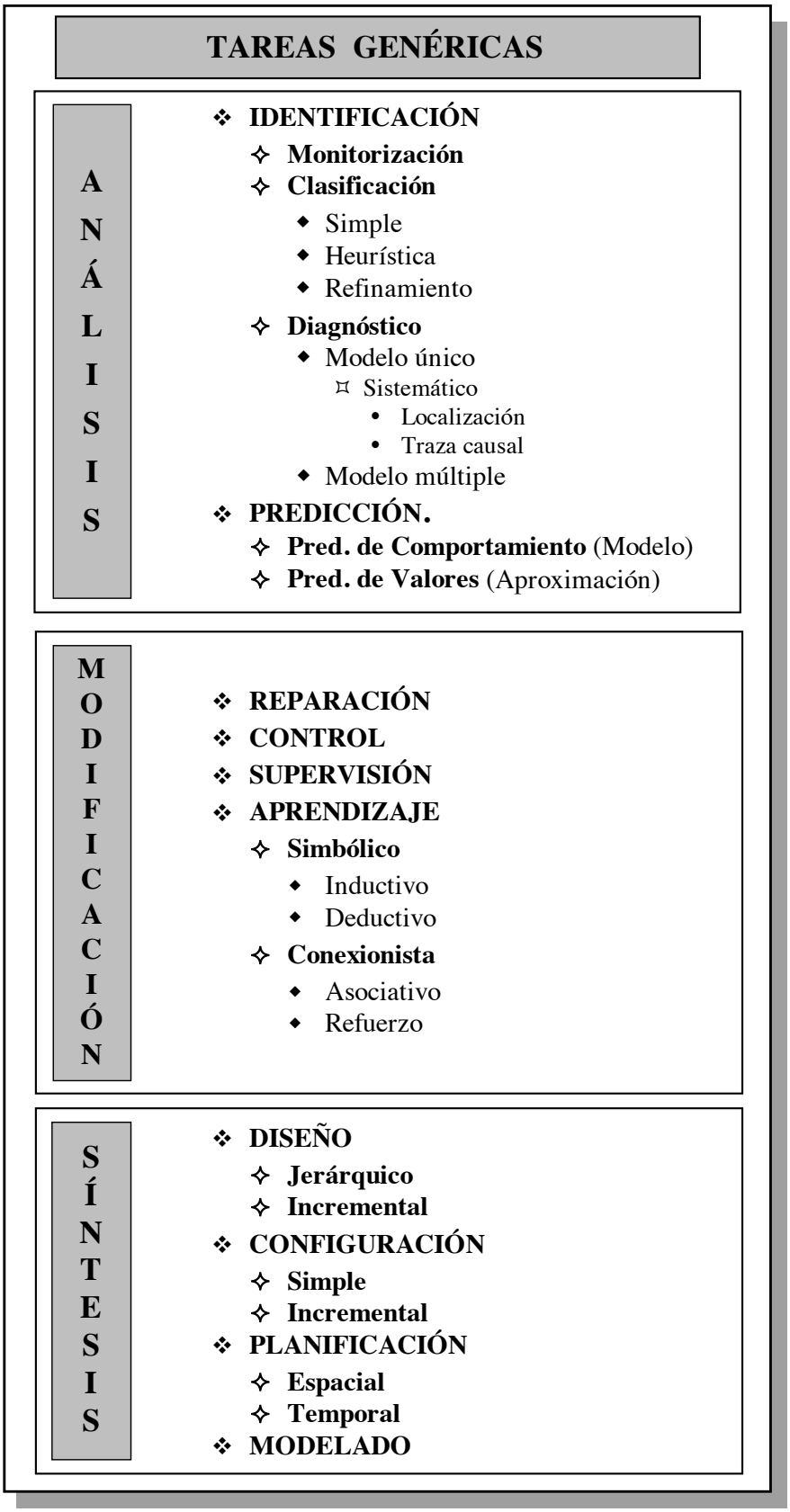


Fig. 4. Relación de tareas genéricas.

Los métodos pueden usarse de forma única o combinada. Es decir, una tarea puede descomponerse usando un sólo método ("establece y refina", "propón-actúa-modifica",o una red neuronal tipo "perceptrón multicapa", etc…) o combinando varios métodos, cada uno para la subtarea para la que resulta más adecuado. Decimos entonces que estamos abordando un problema por técnicas híbridas o multiestrategia.

Al igual que se propusieron bibliotecas de tareas, tambien están disponibles un conjunto de bibliotecas de métodos deductivos, abductivos e inductivos. A nosotros, sin embargo, nos interesa transmitir a nuestros alumnos una idea más fresca, sencilla y coloquial del concepto de método. Son algo así como los "algoritmos" cualitativos formados por una secuencia de ***verbos*** que se extraen de la descripción en lenguaje natural de la tarea y de la posterior reflexión y que nos permiten segmentar esa tarea en otras más elementales que a su vez pueden descomponerse por otros métodos (por ejemplo, segmentar por umbralización) hasta llegar a un nivel de ***inferencias*** elementales, que ya no procede seguir descomponiendo más porque para su aplicación sólo necesitamos rellenarlas con conocimiento del dominio.

4. Pensar que esta lista de componentes reutilizables "pequeñas" a las que hemos llamado inferencias primitivas no son más que los ***subapartados*** de un texto estructurado sobre P&C. Es decir, si estamos realizando un ***preproceso*** o una **extracción** de características en visión de bajo nivel podríamos aceptar que una inferencia primitiva básica es la ***convolución*****,** consistente en pasar una máscara de procesado local (1,1,1; .1,+1; etc..) sobre todo el campo de datos, multiplicando y sumando. Así obtenemos la formulación de todos los operadores locales responsables de la "suavización", "detección de contrastes", etc…).

5. ¿A que llamamos ***conocimiento del dominio***?. En el ejemplo anterior el conocimiento del dominio son los datos crudos, valor de las señales que entran a un operador local de suavización (1,1,1) o de detección de contrastes (-1,+1). Es decir, a los datos, a las señales y, más cerca de los símbolos, a las ***estructuras de datos*** que entran a las inferencias. Nosotros intentaremos usar siempre que sea posible el array, la ***matriz***, como estructura de datos básica en todo el procesado de bajo nivel. Es decir, los sensores nos proporcionarán una matriz de datos, el preproceso la transformará en otra matriz, la umbralización en otra, etc…

En P&C encontramos dos tipos de conocimiento del dominio. Uno que tiene que ver con los datos, que ya hemos mencionado y que es suficiente para el procesado de bajo nivel. Otro, más específico, que tiene que ver con los objetos que deben reconocerse e identificarse en el procesado de alto nivel. Un detector de contrastes nos dibuja el perfil de una llave inglesa pero para "saber" (para que el sistema la reconozca como tal) tenemos que inyectarle ese conocimiento. Con mayor intensidad se necesita este tipo de conocimiento si lo que queremos es reconocer una escena, "comprendiéndola". Por ejemplo, para diagnosticar a partir de una radiografía de pulmón. Recuérdese que al final lo que buscamos son ***descripciones explícitas*** y con significado inequívoco de ***objetos*** *y* ***escenas*** a partir de sus imágenes, que son ***proyecciones bidimensionales*** de un mundo tridimensional.

El ***conocimiento del dominio*** se representa en general en forma de jerarquías de conceptos y relaciones necesarias para especificar lo que un experto sabe y no es genérico. Un concepto viene especificado por su nombre (paciente, síntoma, esquina, puerta, llave inglesa, …), por el conjunto de propiedades que lo definen, por los valores de esas propiedades etc. En general, además, se distingue entre concepto-clase y concepto-objeto

(un elemento de la clase) y el alumno debe recordar que estos **conceptos** pueden tener ***referente físico*** (droga, mesa, pixel, imagen,…) o ser puras ***construcciones teóricas*** introducidas por el experto o por nosotros al modelar su conocimiento, por considerarlas necesarias para facilitar la programación.

Las relaciones entre conceptos también salen de la descripción en lenguaje natural de la tarea y pueden ser de naturaleza causal o de asociación. En general, las encontraremos en verbos del tipo "X es del tipo Y", "Y me sugiere Z", etc… Las más usuales tienen que ver con la herencia ("X es un Y", "Y es parte de Z"), pero no debemos rechazar otras cualesquiera que sean representativas de las relaciones mentales que efectivamente usa el experto humano que nos cuenta cómo realiza una tarea de supervisión o de segmentación de imágenes, por ejemplo. O cómo supervisa un proceso industrial, o cómo decide que son peces, y de qué clase, los que producen esa "sombra" que obtenemos por teledetección mediante sensores remotos. Todo este conocimiento debe describirse de forma explícita para facilitar los relevante en una tarea de interpretación de una escena o de supervisión de un proceso de control, por ejemplo. Este modelado del conocimiento del dominio nos va a permitir establecer relaciones entre los datos de entrada en cada instante y los modelos estructurados creados previamente (la "ontología" del dominio).

Si ya tenemos una idea cualitativa de los conceptos de tarea genérica, método de descomposición, inferencia primitiva y conocimiento del dominio[2], podemos pasar a describir el proceso de reducción. ***Es decir, el paso del modelo al programa***.

La figura 5, ilustra el procedimiento de reducción de los modelos usando métodos simbólicos para la tarea de diagnóstico por "***clasificación jerárquica***". Paralelamente, la figura 6 muestra el procedimiento general, sin hacer referencia a ningún método concreto. La idea básica en todo proceso de reducción de modelos es que el trabajo de desarrollo e implementación de SBC's se ha trasladado a las fronteras del nivel de conocimiento. Por "abajo", en el extremo que conecta con el nivel de los símbolos (el programa), para conseguir pasar del modelo de una tarea genérica y de su método de descomposición asociado a las primitivas de un entorno. Por "arriba", en el extremo que conecta el modelo a nivel de conocimiento con el experto humano, para conseguir adaptar lo que el experto cree que sabe a la estructura de la tarea y de su método asociado. Es decir, para adquirir el conocimiento a través de un conjunto de esquemas verbales que encuentren una correspondencia fácil e inequívoca con las variables y los operadores del modelo (en general un grafo). Como por el extremo de abajo se supone que ya hemos conseguido conectar este modelo abstracto (el grafo) con el nivel de los símbolos, aquí se acaba el proceso de reducción del modelo. Desde el programa hasta la máquina física ya existe un compilador que nos hace el resto del trabajo [1,12].

Así por ejemplo, si hubiéramos llegado a la conclusión que un determinado problema de clasificación de distintos perfiles de alumno puede abordarse mediante la tarea de clasificación jerárquica, nos encontraríamos con un esqueleto tipo árbol en el que los métodos de descomposición de la tarea consisten en la aplicación reiterada de procedimientos de "establecer" y "refinar". Primero se intenta preclasificar una configuración de entrada como perteneciente a una cierta clase y después, se refina dentro de la clase como perteneciente a alguna subclase específica.

---

[2] El alumno que haya cursado SBC's (I), conocerá de sobra estos términos.

En este caso, en la frontera con el nivel simbólico nos preocuparíamos de hacer computables estos verbos. En cambio, en la frontera con el experto humano la preocupación está en averiguar hasta que punto su sistema de pensamiento usa este procedimiento jerárquico para clasificar a sus alumnos y, en caso afirmativo, cómo podríamos representar la información del dominio para que fuera usada de forma natural por ese modelo jerárquico.

Así, el proceso de reducción se puede describir de forma aproximada mediante los siguientes pasos [13]:

1. Tras un primer diálogo con el experto, describir en lenguaje natural la tarea que queremos modelar (diagnóstico, predicción, control, planificación temporal, etc.), eliminando los términos no causales, redundantes y no adecuados para pasar a un modelo computable.

   En el caso del ejemplo empezaríamos obteniendo una descripción sobre cuántos tipos de alumnos podríamos distinguir en un grupo y cuáles son las características que nos permitirían pasar de un tipo general de alumno a cada uno de los subtipos en los que empieza la clasificación. Como resultado de este primer paso también tendríamos algunas "pistas" sobre cuál es el método de clasificación que nos parece más adecuado.

2. Seleccionar o construir el método o combinación de métodos que mejor se adapte para descomponer esa tarea en una red de inferencias hasta alcanzar el nivel de primitivas (inferencias que ya sólo necesitan el conocimiento del dominio).

   Del resultado del paso anterior podemos suponer (ejemplo de la figura 5) que el método más adecuado es la clasificación jerárquica por lo que empezaremos buscando un grafo jerárquico y la forma de describir sus nodos, sus arcos y la forma de pasar, dado un arco y un nodo inicial a un nodo sucesor.

3. Construir la capa de conocimiento del dominio (jerarquía de conceptos, características y valores) especificando las relaciones (aparte de la herencia en las jerarquías) que se consideren necesarias para cubrir la descripción obtenida en (1) y que a la vez sean las que van a necesitar los métodos descritos en (2).

   Una vez seleccionado un método, aparece de forma natural la necesidad de modelar el conocimiento del dominio de acuerdo con ese método. Es decir, siguiendo con el ejemplo de la figura 5, al haber seleccionado la clasificación jerárquica como método de clasificación, tendremos que modelar el conocimiento del dominio mediante una jerarquía de conceptos que representan los tipos de alumnos por los que el método de clasificación va a navegar y el tipo de relaciones entre esos objetos que nos va a permitir hacer efectiva esa navegación.

4. Construir un grafo, o cualquier otro modelo abstracto de computación que nos facilite el paso a un programa a partir de (1), (2) y (3).

   Una vez seleccionado un método (forma de usar los datos de un dominio, enlazándolos en una red de inferencias), es necesario formalizar el modelo correspondiente facilitando el paso a la implementación. Para el método del ejemplo de la figura 5 es evidente que la estructura abstracta intermedia es el grafo jerárquico. En otras tareas y en otros métodos podremos usar otros modelos abstractos.

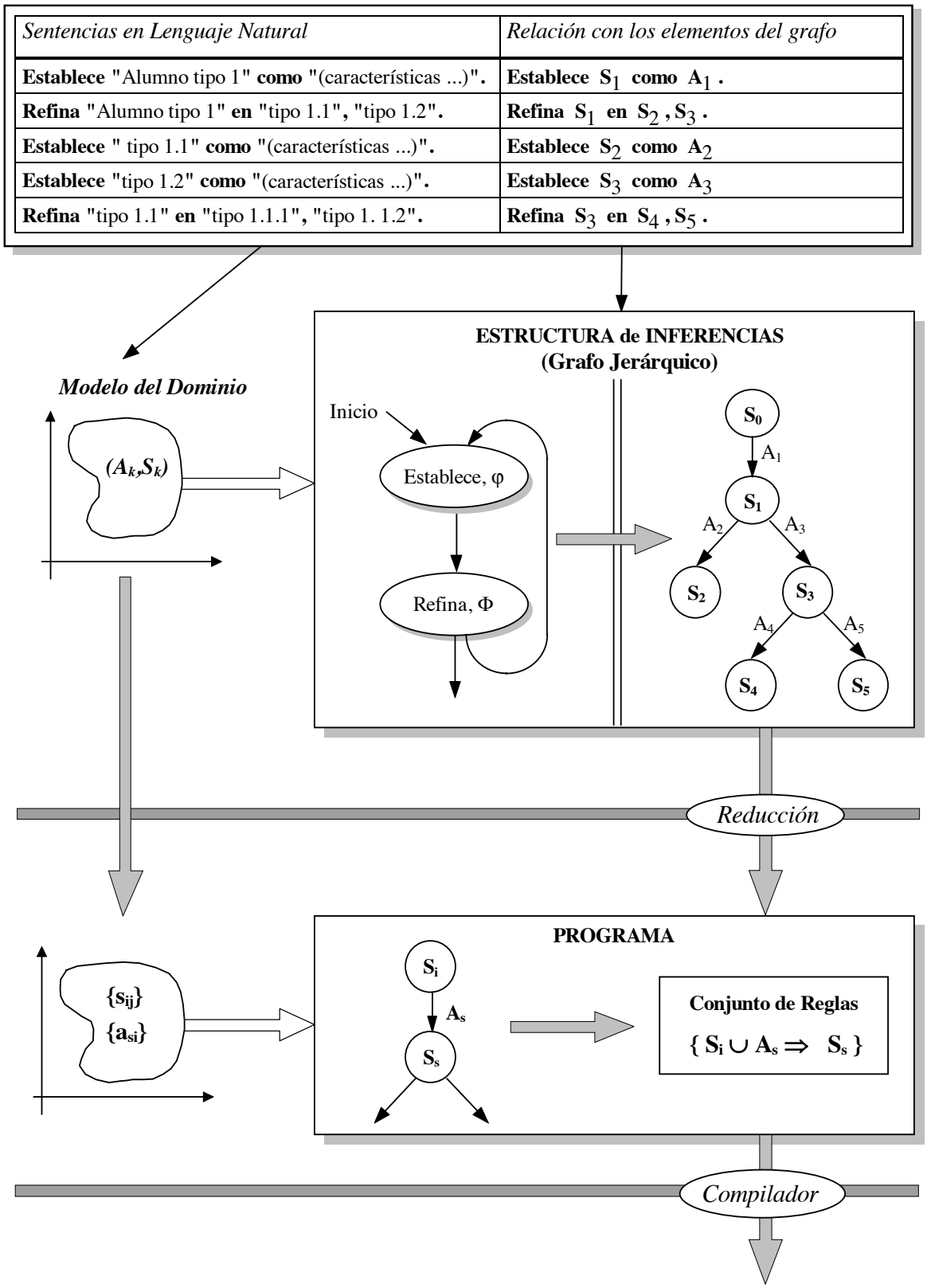


Fig. 5. Ejemplo de reducción de un modelo. Ver descripción en el texto.

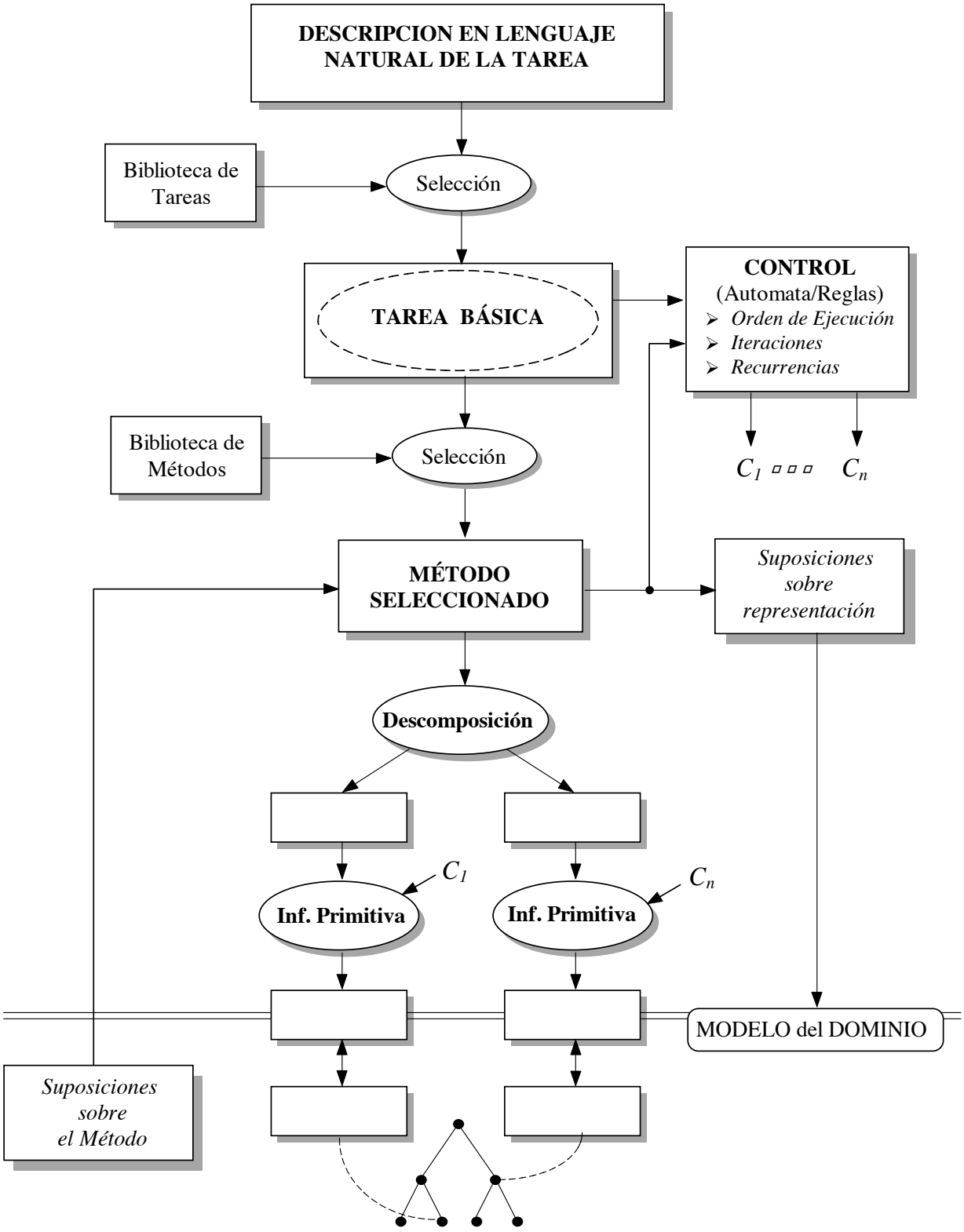


Fig. 6. Fases del proceso de desarrollo de un Modelo de Conocimiento, a nivel de conocimiento y usando componentes reutilizables (métodos, inferencias primitivas y ontologías del dominio).

5. Construir el código para el grafo descrito en (3), completando el proceso de reducción con las tablas semánticas que asocian las descripciones a nivel de conocimiento de los arcos y nodos de (4) (resultados de los pasos (1) y (2)) con las nuevas descripciones de los mismos arcos y nodos aquí en (5). Es decir, lo que de (1) y (2) puede pasar a un programa usando un determinado lenguaje o entorno de programación.

   En el ejemplo de la figura 5 hemos seleccionado la programación por reglas porque se puede deducir de forma prácticamente inmediata, por simple inspección del grafo, a partir de la estructura de inferencias que aparece en la parte intermedia de la figura.

La figura 6 ilustra este proceso de reducción (pasos 1, 2, 3, 4 y 5 que acabamos de describir) del que la figura 5 es un ejemplo sencillo. Obsérvese que lo que hemos intentado hacer en todo el proceso de reducción es encontrar una forma estructurada y sistemática de construir los operadores $\{f,g\}$ del modelo computacional propuesto en la figura 1. Allí quedaba clara también la primera parte del conocimiento del dominio (los espacios de entradas y salidas).

Lo que aquí hemos ilustrado con un ejemplo sencillo y hablando de un problema de clasificación de alumnos debe poderse aplicar al conjunto de tareas y métodos que se van a describir en el resto del material didáctico sobre P&C. Por ejemplo, a las tareas de segmentación, o al reconocimiento de objetos o a la descripción de una estructura de control. Al igual que aquí, una vez seleccionada una tarea, por ejemplo la segmentación, podemos usar varios métodos para llevarla a cabo. Así, podríamos segmentar usando métodos basados en operaciones de umbralizado que hacen uso de información de tipo global acerca de la imagen. Como esta información viene normalmente representada por el histograma, en el desarrollo del método aparecerá como subtarea el cálculo de histogramas. Podríamos también haber seleccionado como método de segmentación la información sobre ***bordes*** o sobre ***regiones***. En estos últimos métodos aparecerá de nuevo subtareas asociadas a la detección de bordes o al crecimiento de regiones etc… Lo que aquí nos interesa señalar es que el método ilustrado en la figura 5 y descrito de forma general en la figura 6 es reutilizable, mutatis mutandis, para modelar todas las tareas computacionales que aparecen de forma natural al analizar los problemas de P&C.

1.5. La Reducción de los Modelos de Conocimiento por la Vía Conexionista

La reducción de un modelo de conocimiento de una determinada tarea por la vía conexionista[3] sigue procedimientos análogos a los de la vía simbólica, con las diferencias asociadas al hecho de que la computación neuronal es una computación modular, de grano pequeño, distribuida sobre varias capas, en las que cada uno de los procesadores realiza una sencilla función analógica no lineal (un sumador espacial o espacio-temporal seguido de una función no lineal tipo sigmoide o tangente hiperbólica que limita la amplitud). Su característica más distintiva es la sustitución parcial de la programación externa completa por el aprendizaje, entendido este como el ajuste en el valor de un conjunto de parámetros (los "pesos").

El único carácter distintivo de la reducción mediante redes neuronales artificiales está en el hecho de que el modelo estructural es invariante. Es decir, si por cualquier razón

[3] Para el alumno que vaya a estudiar o haya estudiado la asignatura de SBC'II (Redes Neuronales), este apartado le será de sobra conocido.

elegimos un método conexionista para resolver un problema, ya sabemos que terminaremos usando una red multicapa en la que se asocia una capa a cada una de las inferencias básicas en términos de las cuales se descompone la tarea, de acuerdo con su modelo a nivel de conocimiento. El método neuronal admite que la tarea que queremos resolver se puede formular en términos de un ***clasificador multicapa*** en el que cada capa se encarga de calcular una inferencia. En la construcción del modelo a nivel de conocimiento hemos usado todo lo que sabíamos sobre el problema dando lugar a un primer esquema, como el que aparece en la figura 7, con la siguiente segmentación en ***subtareas***:

1: ***Preproceso, Normalización y Etiquetado***
2: ***Cálculo de descriptores***
3: ***Métricas. Cálculo de distancias a clases***
4: ***Diagnóstico***
5: ***Valoración***
6: ***Algoritmos de Aprendizaje***

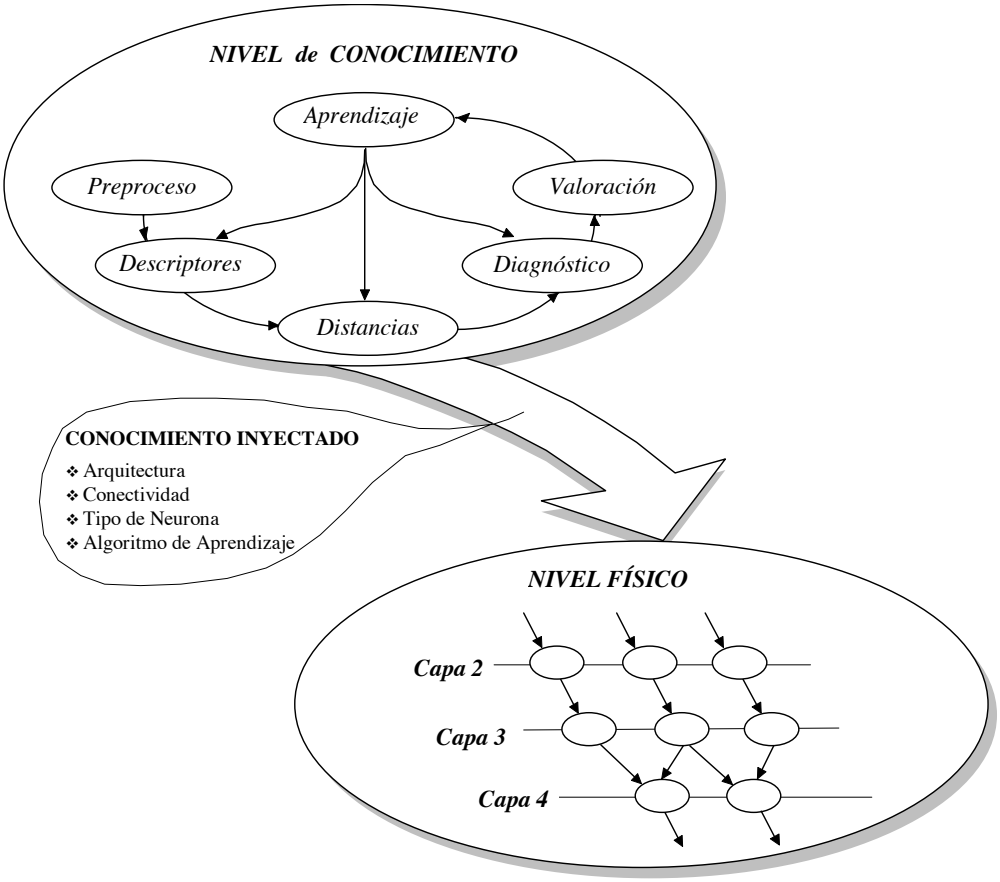


Fig. 7. Reducción del nivel de conocimiento al nivel físico en computación neuronal para una tarea de clasificación.

A cada una de estas subtareas le asociaremos después *una capa* de la red, un tipo de neurona y un algoritmo de aprendizaje dependiendo, como siempre, del conocimiento del que disponemos "a priori" sobre el dominio de conocimiento (imágenes en meteorología, imágenes médicas, voz, matrículas de coches, etc…).

1.5.1. Preproceso y Espacio de Entradas.

Es decir, preparación de las señales (variables en general) que van a constituir el ***espacio de entradas*** de nuestra red neuronal. Aquí ya hay una gran cantidad de conocimiento inyectado. Si elegimos variables poco relevantes e intervalos poco adecuados, la red no podrá clasificar adecuadamente.

Cada una de las variables resultado del preproceso se convierte en una ***línea etiquetada*** de entrada a la red, con su correspondiente ***tabla de semántica***. La red maneja números y su significado siempre depende de las tablas que quedan en el *DO* a nivel de conocimiento. A partir de aquí la red "no sabe" si lo que está clasificando son melones o síntomas clínicos.

1.5.2. Espacio de Salidas (Clases).

El resultado de la clasificación aparece en las salidas de las neuronas de la última capa de forma que hay una cierta codificación implícita. De nuevo la salida de cada neurona es una *línea etiquetada* y la que se dispara nos dice que esa es la clase a la que pertenece la configuración que se está clasificando en ese momento.

Así, si consideramos necesarias *n* clases, necesitamos *n* neuronas en la última capa, cada una con su ***tabla de semántica***. Es evidente que las redes probabilísticas o borrosas pueden incluirse sin dificultad si aceptamos que las distintas salidas (y_1, …, y_n) toman valores en el intervalo [0,1] y consideramos el perfil de la "curva" de respuestas como una medida de la ***probabilidad*** [$y_j(t)$ = p(clase j-ésima)], de que la configuración de entrada pertenezca a la clase j. O bien, [$y_j(t) = \mu_j$], como la posibilidad o grado de compatibilidad de la configuración de entrada con la clase j-ésima.

Obsérvese de nuevo la cantidad de *conocimiento inyectado* en el diseño de la red al limitar, especificar y etiquetar las distintas *clases de equivalencia* que constituyen su espacio de salidas.

1.5.3. Extracción de Características.

Si ya tenemos especificadas las entradas y salidas de la red, el siguiente paso es especificar la arquitectura multicapa, el tipo de neurona seleccionado para cada capa y ser algoritmo de aprendizaje más adecuado para cada neurona dentro de cada capa. Es evidente que para que una tarea merezca la solución neuronal el problema debe ser ***segmentable***, parte de su solución debe poderse asociar a la ***conectividad*** y es justificable la necesidad de ***autoprogramación*** (aprendizaje) en tiempo real porque el medio en el que va a trabajar la red es cambiante y sólo parcialmente conocido.

Pasamos entonces a ***modelar*** cada una de esas ***subtareas***, empezando con el ***cálculo de los descriptores***. Un problema fundamental del reconocimiento de caracteres es la enorme dimensión de los espacios de entradas. En general tenemos muchas variables cuyo proceso no podemos abordar de forma eficiente en tiempo real. Por consiguiente, es crucial hacer una *selección* y *reducir* la dimensionalidad del espacio de entradas.

1.5.4. Métricas. Cálculo de Distancias a Clases.

Si ya tenemos una capa neuronal que calcula un conjunto de descriptores de la entrada (piénsese en una imagen, por ejemplo), el siguiente paso es determinar las distancias de estas configuraciones de descriptores a los valores representativos de cada una de las clases. Lo que buscamos es una capa de neuronas que divida el espacio de propiedades en tantas regiones como clases, cada una de las cuales vendrá representada por su "vector prototipo" o "centro de gravedad" de la nube de puntos que la constituyen.

1.5.5. Diagnósticos.

El siguiente paso es usar una capa para que decida sobre la clase a la que "creemos" que pertenece la configuración. Esto se puede realizar de varias formas:

- ✧ Selección del valor máximo: procedimientos competitivos.
- ✧ Procesos cooperativos.
- ✧ Interpretación borrosa o probabilística del resultado final.

Tras una etapa inicial de competición entre la *IA* conexionista y la *IA* simbólica, y reconocidas sus limitaciones en ambos casos, el estado actual del conocimiento sugiere la conveniencia de integrarlas de forma que al modelar el conocimiento en el *DO* mediante una estructura de tarea y métodos se decida qué tareas aconsejan una implementación mediante métodos simbólicos y cuáles una implementación mediante métodos concexionistas, incluyendo el diálogo y la cooperación entre ambas implementaciones de tal forma que podamos hablar de ***Sistemas Expertos Conexionistas*** y ***Redes Neuronales Basadas en Conocimiento*.**

A lo largo de estos cinco primeros apartados del tema he procurado mostrar el carácter normal de la actividad en *IA*. Normal en el sentido que no se hacen cosas distintas de las que otros profesionales de la computación:

> ***Dialogar con las fuentes del conocimiento, modelarlo, buscar un puente a través de un modelo abstracto intermedio (autómatas, grafos, lógica) y enlazar finalmente las primitivas del modelo con las primitivas de un entorno de programación.***

Lo peculiar o característico en *IA* procede de: (I) la naturaleza del conocimiento que se quiere computar, en general no analítico, poco claro, impreciso, equívoco y contradictorio y siempre basado en descripciones en lenguaje natural. (II) Los métodos usados en la solución, en general más próximos a búsquedas con heurísticas que a algoritmos, en el sentido usual (aunque finalmente, las búsquedas son también algorítmicas). (III) la importancia que se da al "aprendizaje", tanto simbólico como conexionista. y (IV) la dificultad intrínseca de las funciones humanas que pretende emular (visión, movimiento, …) que no sólo necesitan duplicar la "mente", sino también el "cuerpo", sin el cual la propia mente humana no sería tan eficaz para adaptarse y controlar el medio.

1.6. Modelado de la Percepción a Nivel de Conocimiento

Ya hemos comentado que la tarea de percepción es muy amplia, por lo que necesita descomponerse en otras subtareas más elementales hasta llegar al nivel de inferencias primitivas. En la figura 1.2 del texto de *Introducción a la IA* [4] que aquí repetimos por comodidad como figura 8, se establece la primera gran distinción entre procesado de bajo y alto nivel.

"La idea de partida es que este paso se produce cuando es necesario dar un salto cualitativo en el conocimiento que es necesario inyectar desde el exterior de un sistema para comprender el significado del proceso. Este salto se da en visión al pasar del ***nivel bajo*** (transductores, preproceso, extracción de características locales y segmentación) al ***nivel alto*** en el que se define el espacio de objetos y se realiza el reconocimiento de patrones y la interpretación del *significado* de una escena."

"En el procesado de bajo nivel prácticamente toda la información está en la imagen. El transductor sigue una ley física, en general de tipo logarítmico. Por ejemplo, pasa de voz o de imagen a una señal eléctrica continua y variable con el tiempo *v(x,y;t)*, usando un amplificador o una cámara de vídeo. A su vez, esta señal analógica se digitaliza mediante un conversor analógico-digital que cambia drásticamente su forma pero mantiene la información. El segundo paso es el *preproceso* que extrae características locales o integrales de naturaleza analítica (derivadas espacio-temporales, transformadas de Fourier, etc...) que no exigen conocimiento complementario para ser entendidos por un observador. Este preproceso produce una descripción complementaria de la señal inicial que facilita el posterior proceso de clasificación. Aunque hemos dicho que aquí no es necesaria la inyección de conocimiento a nivel explícito, sí que existe de forma implícita a través de la selección del conjunto de características que consideramos más adecuadas para el reconocimiento de la voz o de una determinada familia de imágenes."

"La etapa final del procesado de bajo nivel es el reconocimiento de formas basado en la definición analítica de distancia entre el valor que toman las propiedades usadas para describir la imagen y los valores correspondientes a esas variables en un conjunto de patrones. La salida de este nivel es la etiqueta del patrón al que más se aproxima la descripción. Aquí ya hay un primer paso de semántica al definir las clases de equivalencia y las métricas usadas para discriminar, aunque el salto cualitativo está en el segundo nivel."

"En el procesado de alto nivel (percepción), nos hace falta recurrir a la inyección de conocimiento externo específico del dominio para dar significado a las estructuras de datos y a los procesos porque su sentido sólo queda claro para quien posee ese conocimiento del dominio y en ningún caso es evidente a partir de las entidades del nivel simbólico o del nivel físico (en el caso del conexionismo). El punto frontera es la definición del espacio de objetos en el modelo del medio. Dos patrones con la misma etiqueta a nivel sintáctico pueden representar entidades de la escena totalmente diferentes a nivel semántico. Este segundo nivel, de comprensión de imágenes, es responsabilidad de la *IA*."

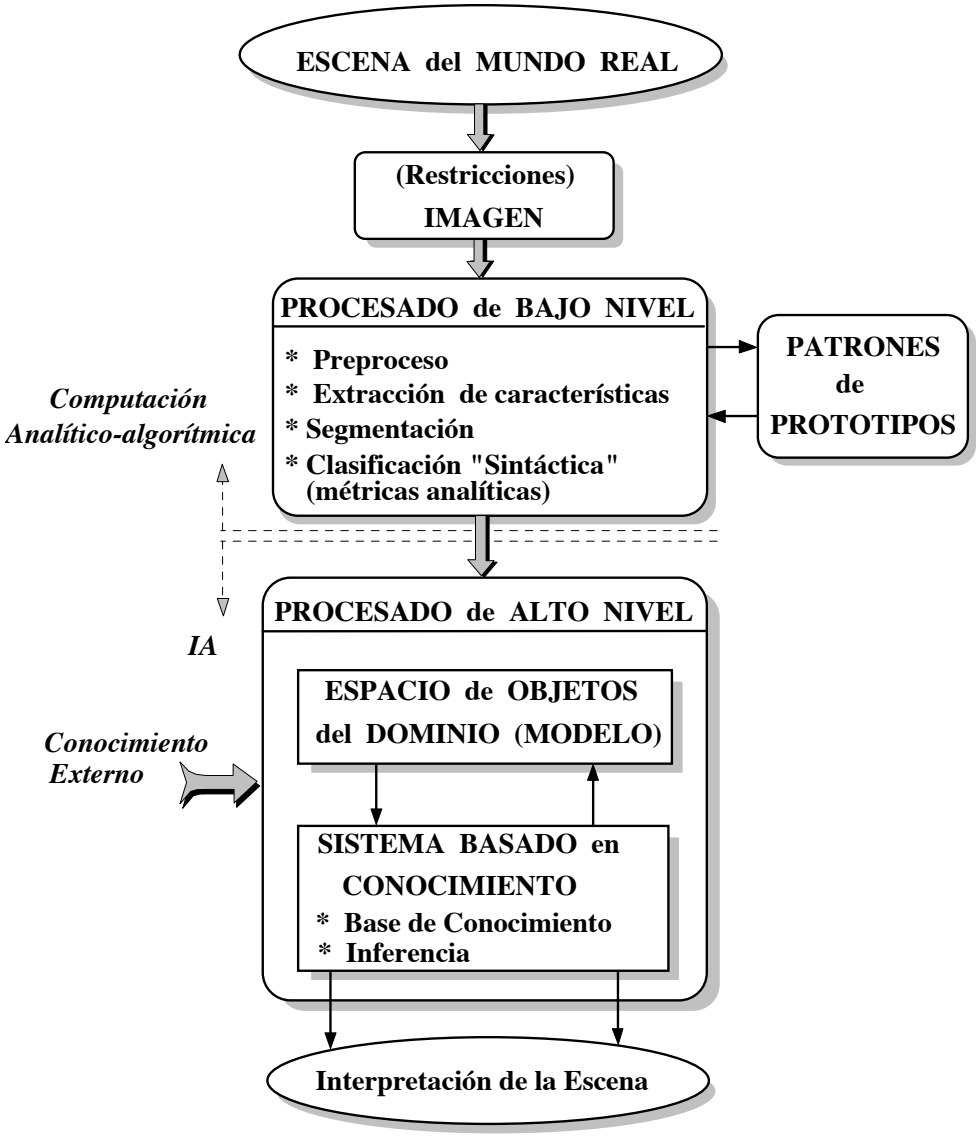


Fig. 8. Representación esquemática del proceso de visión artificial, distinguiendo el procesado analítico de bajo nivel del procesado de alto nivel, basado en conocimiento y propio de la *IA*.

"Comprender una imagen es asociarle la escena del mundo real que representa de acuerdo con un modelo del medio. Pensemos por ejemplo en el proceso que realiza un médico para interpretar una radiografía y emitir un informe diagnóstico. Es claro que en la imagen se encuentran elementos identificables a nivel analítico (contrastes, determinadas bandas del histograma, etc...). Sin embargo, el diagnóstico final (la comprensión de la imagen) es una consecuencia del *modelo* de los "objetos del dominio" que el médico posee sobre las configuraciones normales o patológicas de la anatomía del órgano cuya imagen se está procesando y este conocimiento no está en la imagen." (Tomado de J. Mira y A.E. Delgado, ref. 4).

Si observamos con cierto detenimiento la figura 8 y la descripción previa vemos cómo aparecen las subtareas en las que puede descomponerse el procesado de imágenes. Las repetimos con ligeras modificaciones:

BAJO NIVEL

1. Representación sensorial y digitalización.
2. Preproceso (de pixel, local y de restauración).
3. Segmentación (por umbralización, detección de bordes, comparación con patrones, etc…)
4. Descripción de objetos en términos de un conjunto de características que faciliten su clasificación (area, forma, centro de "masas", …).
5. Reconocimiento de objetos (clasificadores estadísticos, neuronales o híbridos).

ALTO NIVEL

6. Interpretación de la escena.

Cada una de estas subtareas puede a su vez descomponerse por uno o más métodos o por la combinación de varios. Así, podemos ***segmentar*** basándonos esencialmente en un proceso de ***umbralización*** de la señal con distintos valores de umbral y/o con umbralización adaptiva que aplica en cada zona el valor de umbral que mejor se adapta. También podemos segmentar usando el resultado de filtrar la imagen pasa alta, pasa baja y pasa banda o usando el color. Finalmente, podemos combinar estos métodos y usar **conocimiento del dominio** para optimizar el proceso de segmentación. Es decir, usar lo que sabemos sobre el tipo de imágenes que estamos procesando (de medicina, planos de ciudades, rutado de circuitos electrónicos, caras, etc…) para ayudarnos en la tarea de segmentación.

Cada subtarea tiene como entrada una o varias estructuras de datos y produce a la salida una o más estructuras a las que hemos llamado previamente "roles" (papeles que representan los datos en cada inferencia). La figura 9 muestra un ejemplo de esquema jerárquico que engloba las distintas subtareas que hemos mencionado y menciona, de forma cualitativa, las estructuras de datos que las enlazan. No nos importa tanto el detalle de la jerarquía como la conveniencia de partir de esquemas de este tipo al comenzar el modelado conceptual de cualquier subtarea.

Como es obvio, no debemos desarrollar aquí todas las subtareas. Ese es el propósito del resto de los capítulos. Sin embargo, sí que queremos ilustrar el proceso de modelado seleccionando una ***subtarea*** y, dentro de ella, un ***método*** de descomposición para ilustrar el procedimiento general. Por razones pedagógicas, vamos a comentar ***la extracción de características*** mediante filtros descritos en el dominio del tiempo por operadores que realizan la convolución de la imagen con un núcleo de ponderación, con una máscara, Esta máscara es el resultado de la descripción de la función del operador en diferencias finitas.

Fig. 9. Una jerarquía de subtareas en el procesado de bajo nivel y su preparación para el posterior proceso de "comprensión" (tomada de G. Jähne, "Digital Images Processing").

Para descargar la descripción de toda dependencia con el formalismo matemático, usaremos reglas sencillas para describir el proceso[4].

1.7. La Convolución como Método

1.7.1. El Filtrado Analógico

La idea de convolución está asociada en física al proceso de medida de una magnitud física como suma ponderada de un conjunto de valores de la señal por un conjunto de pesos que representan el "metro". La importancia del operador de convolución en el procesado de señales es consecuencia de su poder para representar el comportamiento de cualquier sistema lineal en el dominio del tiempo. Si llamamos *x(t)* a la entrada, la respuesta del sistema es de la forma,

$$y(t) = \int_0^t x(t) \cdot g(t-\tau) \cdot d\tau$$

Es decir, para calcular un valor de la salida, y(t), necesitamos conocer el valor de la entrada, x(t), en todos aquellos instantes previos ($\tau < t$) en los que el núcleo de ponderación, $g(t-\tau)$, tiene valor significativo. Esta función, $g(t-\tau)$, es la función ponderatriz, que se obtiene al excitar al sistema con un impulso y observar cómo evolucionan sus modos naturales de volver al equilibrio.

Una vez conocida $g(t-\tau)$, sabemos todo sobre la dinámica de ese sistema. Dada cualquier entrada $x(\tau)$, vamos multiplicando punto por punto $x(\tau)\cdot g(t-\tau)$ y después sumamos sobre τ, tal como se ilustra en la figura 10. Es decir, calculamos el **área** bajo la curva producto. Como para saber el valor de *y(t)* en un punto, tenemos que conocer el valor de *x(t)* en todo el rango en el que $g(t-\tau)$ no es nula, los matemáticos nos dicen que $y(t_1)$ es un funcional. Para calcular el valor de la salida del filtro en otro instante, $y(t_2)$ tenemos que **desplazar** el núcleo de ponderación, $g(t-\tau)$, poniendo su centro debajo de t_2 y volviendo ahora a **multiplicar** punto a punto y **sumar** y así para todos los valores de t= $t_1, t_2, t_3, \ldots, t_n$.

Queda entonces claro el método general de usar el operador de convolución para extraer propiedades en una señal, *x(t)*, (para "medir" su participación en esa propiedad). Lo que detecta un núcleo $g(t-\tau)$ en una señal *x(t)*, es *y(t)* que mide lo que *x(t)* tiene de $g(t-\tau)$. Si *g* es un núcleo suave y de un sólo signo, sacaremos el contenido de baja frecuencia de la señal *x(t)*, la "suavizaremos". Si por el contrario $g(t-\tau)$ es un "núcleo en diferencias", con partes positivas y negativas, sacaremos el contenido en alta frecuencia, apareciendo muy reforzados en y(t) los cambios de pendiente que tenía *x(t)*. Es decir, ahora $g(t-\tau)$ actúa como un derivador que acentúa los contrastes y nos facilita su detección posterior

[4] Evidentemente, para el lector que conozca el formalismo usual en Sistemas lineales, la integral de convolución y las transformadas de Fourier y Laplace, esta descripción cualitativa, usando reglas y dibujos no lo será necesaria.

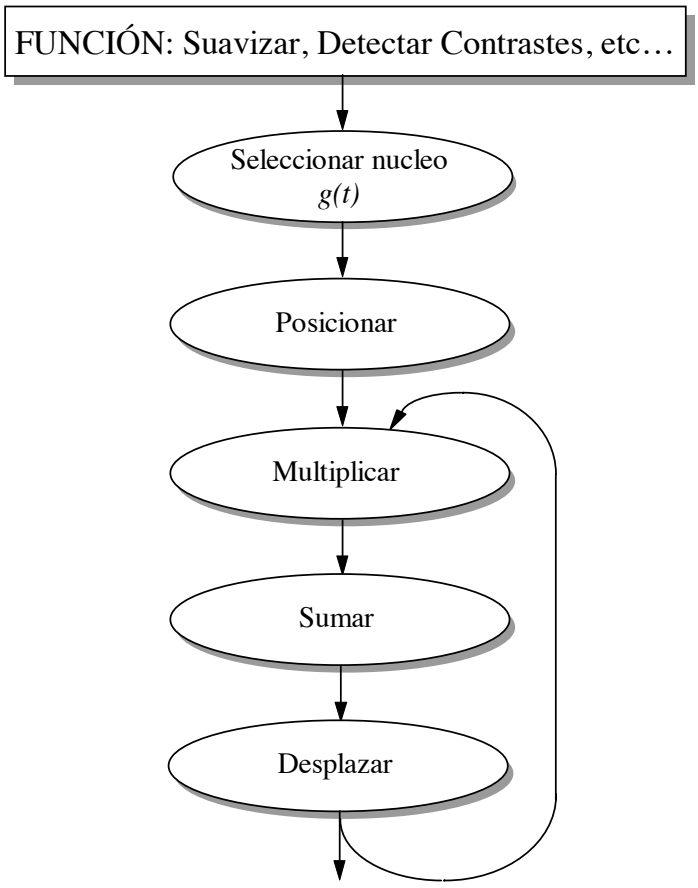


Fig. 10. La convolución como método. En la parte superior se representa la señal de entrada, *x(t)*; el núcleo de ponderación, *g(t)*, su versión desplazada a $t = \tau$ y pasada por un espejo, el producto $x(t) \cdot g(t-\tau)$ para todos los puntos en los que *g* tiene valor y el área bajo la curva producto, $y(\tau)$. En la aprte inferior describimos el método: selección de núcleo, posicionar, desplazar, multiplicar, sumar, desplazar, …

La figura 11 ilustra el proceso de convolución para núcleos integradores (sumadores), que suavizan la imagen y para núcleos derivadores (que restan valores propios de valores de los vecinos) y que refuerzan los contrastes. La idea importante que nos gustaría transmitir es que el núcleo lleva en su estructura local la información sobre la propiedad (local) que detecta en la imagen. Mirando un núcleo, debemos saber predecir la propiedad que detecta. Inversamente, partiendo de una descripción en lenguaje natural de lo que queremos que nos detecte un núcleo, deberíamos saberlo escribir.

Cuando pasamos de este ejemplo unidimensional al tratamiento de imágenes las operaciones de filtrado generan para cada pixel de salida un nivel de gris que es función del nivel de gris del pixel que ocupa la misma posición en la imagen de partida y del nivel de gris en un área local que cubre un conjunto de pixels vecinos. Así, el equivalente al filtro pasa baja de la parte (a) de la figura 11 serán los filtros de suavizado como los de la media o los
k-vecinos, etc. Algunos de estos filtros tiene especial interés por su habilidad en preservación de bordes, es decir, la facultad de reducir el ruido presente en una imagen y a la vez ser capaces de retener, sin eliminar ni difuminar, las fronteras o bordes entre zonas diferenciadas de la imagen.

El equivalente en imágenes al sencillo filtro unidimensional de la figura 11(b) es la detección de contrastes espaciales, temporales y espacio-temporales en cualquier fase de la descripción de una imagen. Se usan para encontrar fronteras de regiones, dando por supuesto que dentro de una región hay una cierta homogeneidad en el valor de iluminación. Su expresión general tiene que ver con el operador ***gradiente*** que en cada punto detecta la dirección de máximo crecimiento y el valor de ese crecimiento. Para aquellos alumnos que posean conocimientos sobre ecuaciones diferenciales y transformaciones tipo Fourier y Laplace diremos que la función de un filtro se describe a partir de la forma de su función de transferencia, transformada de la función de ponderación. Sin embargo, como damos por supuesto que estos conocimientos no son necesarios en nuestros alumnos, pasaremos cuanto antes (tras la descripción general de filtros recursivos y no recursivos) a la descripción aproximada de estas operaciones de filtrado sustituyendo las ecuaciones diferenciales por ecuaciones en diferencias finitas. Es decir aproximando las derivadas por restas de valores en puntos próximos (entre un pixel y su vecino).

La forma más general de representar un filtro es considerar que, independientemente del tipo de núcleo, g, puede incluir una o más de las siguientes posibilidades.

a) Filtrado espacial no recursivo.
b) Filtrado espacial recursivo.
c) Filtrado temporal no recursivo.
d) Filtrado temporal recursivo.

En las figuras 12 y 13 se resumen estas posibilidades. En 12(a) se muestra (para un núcleo en diferencias de gausianas), el caso espacial no recursivo en el que para obtener la respuesta en un punto del eje x, tomamos en cuenta los valores de los vecinos. Si, además, consideramos también las respuestas de cada punto de cálculo y de los puntos vecinos, tenemos un filtro espacial recursivo (caso b).

(a)

(b)

Fig. 11. Ilustración del efecto del filtrado pasa alta y pasa baja en una señal unidimensional. (a) Núcleo pasa baja "suavizador" que sustituye valores de un pixel por valor medio en el entorno. (b) Núcleo en diferencias, "derivador" que acentúa los cambios de pendiente.

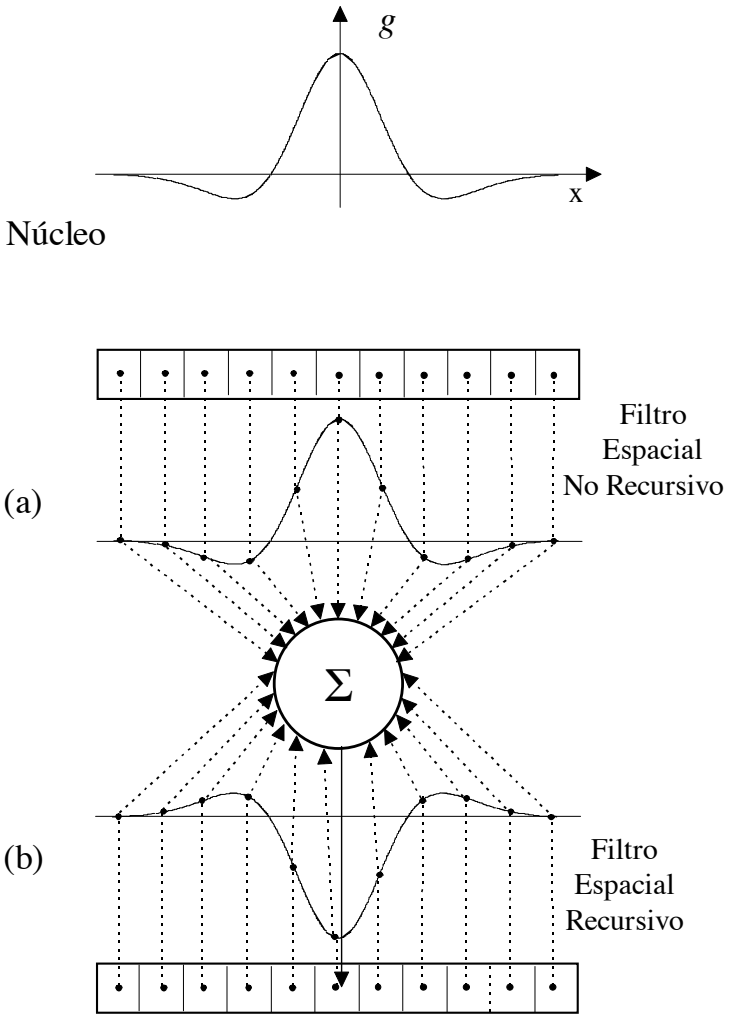


Fig. 12. Forma general de los filtros espaciales analógicos. En la parte superior se muestra la forma de la función de ponderación, tipo centro excitador y periferia inhibidora. En (a) se representa la configuración espacial no recursiva. Si le añadimos al filtro la parte (b) se convierte en un filtro espacial recursivo, que realimenta también una suma ponderada de su respuesta y de la respuesta de sus vecinos.

En la figura 13 se representan los filtros temporales recursivos y no recursivos. Ahora no consideramos lo que ocurre a la vez en varios puntos próximos de una imagen, sino lo que ocurre en una única coordenada durante instantes sucesivos de tiempo. En la parte (c) se representa el filtrado no recursivo, en el que la respuesta, $y(t)$, en un momento dado, es función de las entradas, $x(t)$, en ese instante y en instantes anteriores ponderadas por el núcleo, $g(t-\tau)$. En el caso (d), representamos un filtro temporal recursivo en el que no sólo se ponderan las entradas, sino que también entran a sumar convenientemente ponderadas las respuestas en instantes anteriores. Esto significa que el filtro debe poseer una memoria local (tipo FIFO) en la que se van almacenando las respuestas en cada instante colocándolas en la dirección más alta de la memoria y se van "empujando" hacia "abajo" las respuestas en los instantes anteriores. En cada nuevo ciclo de cálculo, se lee el contenido de estos registros y se multiplican por el núcleo. Después, se suma con las contribuciones de la entrada para producir el nuevo valor y así sucesivamente.

La combinación de las figuras 12 y 13 genera cualquier superficie de ponderación en el espacio-tiempo y, por consiguiente, es el modelo más general de filtrado lineal.

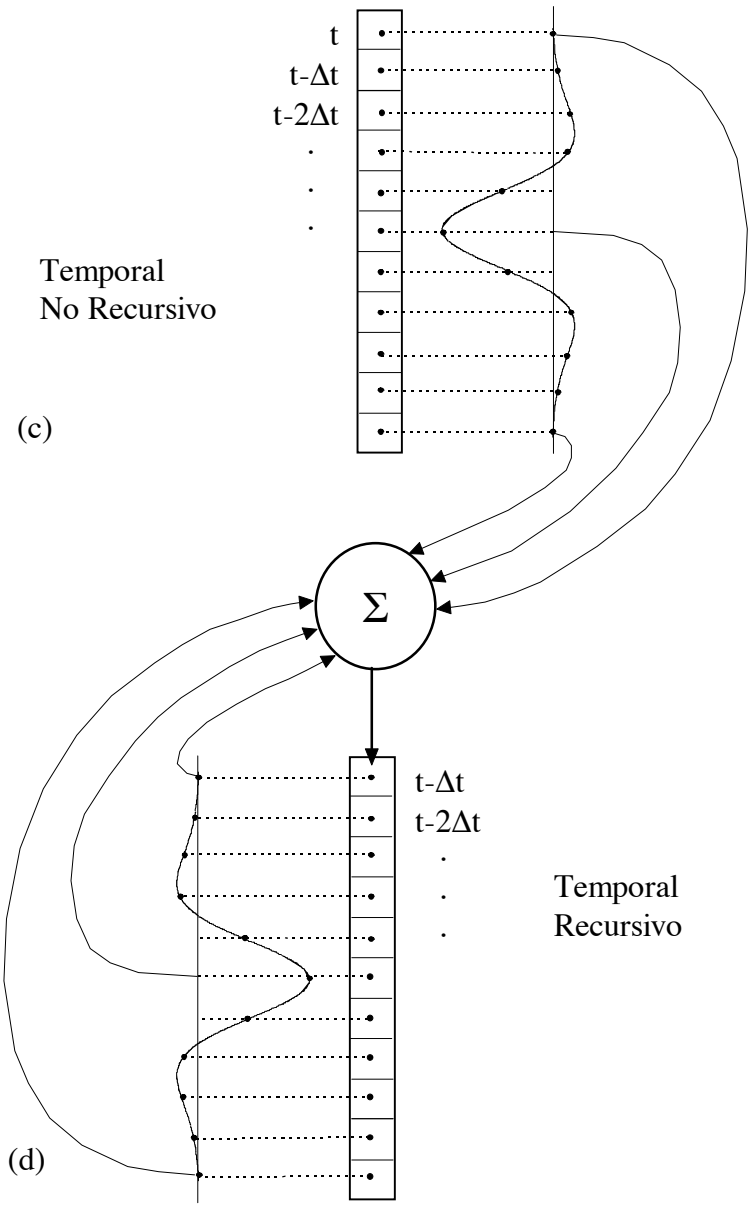


Fig. 13. Forma general de un filtro temporal recursivo. En la parte superior (c) aparece la componente no recursiva en la que se ponderan sólo las entradas en el instante actual (t) y en varios instantes anteriores. En la parte inferior (d) se le añade la realimentación ponderándose también las respuestas en instantes anteriores.

1.7.2. La Formulación Local del Filtrado Digital

La imagen, una vez digitalizada, tiene el aspecto de una matriz NxN (por ejemplo, 1000x1000) y el núcleo se convierte en otra matriz numérica nxn (por ejemplo, 3x3), de forma que ahora la convolución se representa mediante el producto coordenada a

coordenada de la matriz núcleo, $G(\alpha,\beta)$, por la matriz de entrada $X(x,y)$, para producir una nueva matriz de salida $Y(x,y)$. el procedimiento, sin embargo, es el mismo que en el caso analógico. Primero posicionamos el núcleo G, después multiplicamos y sumamos, después normalizamos y escribimos el valor de la respuesta bajo las coordenadas centrales de G y, finalmente, desplazamos y repetimos el proceso.

El origen de las expresiones numéricas y con pocos valores de las matrices de ponderación, g, que caracterizan el comportamiento digital de un filtro está en el muestreo de la señal analógica de partida, g(t), despreciando el valor de las muestras a partir de un cierto umbral. También pueden obtenerse partiendo de la ecuación diferencial que describe el comportamiento dinámico del filtro y simplificando las derivadas que se reescriben ahora como incrementos. Se obtiene así, la expresión en diferencias finitas "equivalente" a la ecuación de partida y de esta expresión nace el núcleo de ponderación.

Veámoslo en un caso sencillo. Supongamos que queremos encontrar la matriz de ponderación digital correspondiente a un filtro analógico que viene descrito por la ecuación de segundo orden que todos estudiamos en mecánica (fuerza = masa x aceleración = masa x derivada de la velocidad).

$$f = -Kx = m\frac{d^2x}{dt^2} \approx m\left[v(t) - v(t-1)\right], \quad \text{con} \quad \Delta t = 1$$

aproximando $v(t)$ como $x(t)$-$x(t-1)$ tenemos la ecuación en diferencias finitas

$$f = -Kx = m\left[x(t) - 2x(t-1) + x(t-2)\right]$$

Es decir, para obtener el valor local en una determinada posición y en un determinado momento, $x(t)$, sólo necesitamos conocer su valor en esa posición en el instante anterior $x(t-1)$ y en dos instantes anteriores, $x(t-2)$. Estos dos números junto con el coeficiente $1/m$, definen la computación local (figura 14). Obsérvese que para sintetizar este filtro sólo hacen falta dos retardos (una memoria local que guarda el valor de la señal en dos intervalos sucesivos de tiempo). La clave está en los retardos y en la proporción $(2, -1)$ con la que se ponderan las muestras. Observen además que este filtro se convertiría en un oscilador si $f=-Kx$, lo que exige un lazo de realimentación local adicional para convertir en entrada un valor proporcional y de signo opuesto (-x) al de la respuesta.

Al igual que la máscara [2, -1], resuelve una ecuación de segundo orden, todas las máscaras del tipo:

$$\begin{bmatrix} 1 & 1 \end{bmatrix}, \quad \frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \quad ....$$

son ***suavizadores*** ya que calculan valores medios.

Análogamente, como hemos sustituido las derivadas por restas sobre pixels vecinos todas las máscaras de la forma [-A, B, -A] detectan contrastes y sirven para reforzar las zonas en las que una imagen presenta bordes. Por ejemplo,

$$\begin{bmatrix} 1 & -1 \end{bmatrix},\ \begin{bmatrix} -1 & 1 \end{bmatrix},\ \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix},\ \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix},\ \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix},\ \ldots.$$

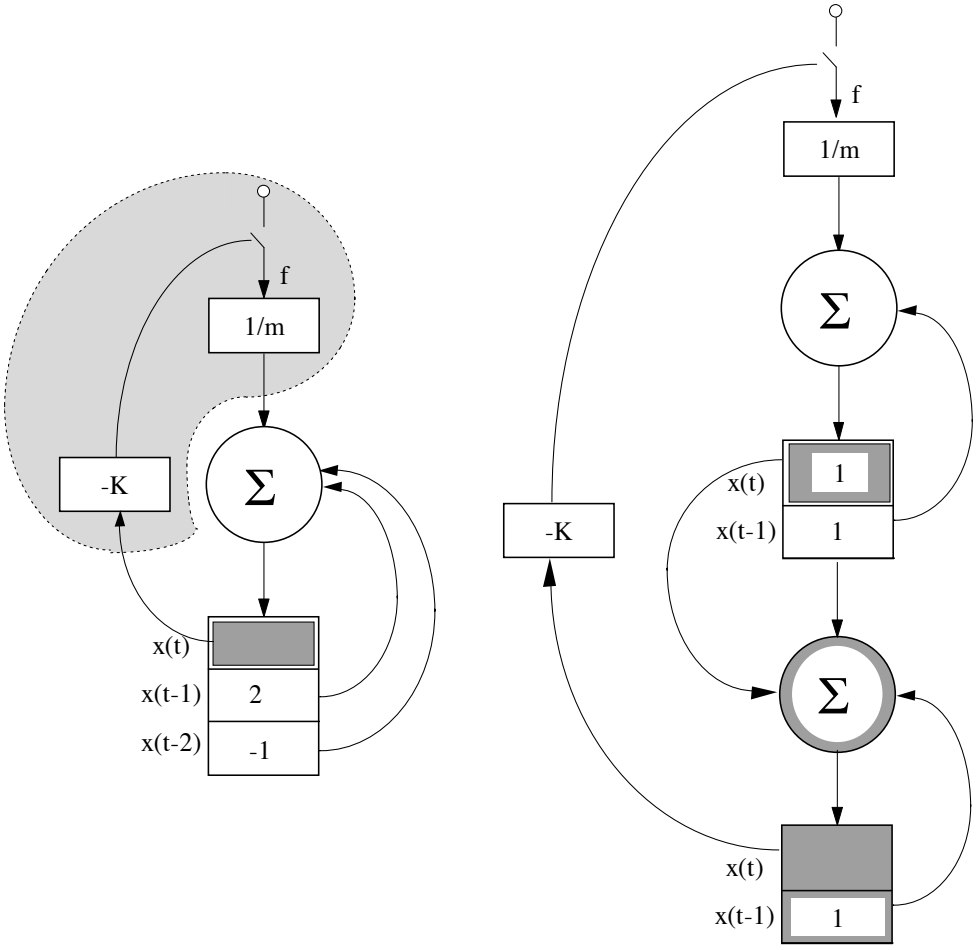


Fig. 14. Ilustración de la expresión local de la ecuación diferencial de 2º orden que da origen al núicleo de ponderación [2, -1]..

Si hiciéramos un análisis del operador Laplaciana sustituyendo sus derivadas por incrementos obtendríamos una máscara de la forma:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

y de la misma manera podríamos obtener la máscara correspondiente a cualquier otro operador diferencial. Afortunadamente para nuestros alumnos es más intuitivo y sencillo pensar directamente en términos de matrices cuadradas de pocos elementos que describan operaciones locales adecuadas a la tarea que queremos resolver y escribirlas directamente, independientemente de que corresponda o no a expresiones en diferencias finitas de determinadas ecuaciones diferenciales.

Afortunadamente. también el uso de la convolución como método en las tareas propias del tratamiento digital de imágenes, puede formularse y programarse sin hacer referencia explícita al cálculo diferencial. Basta pensar en operadores locales y el resultado de su desplazamiento sobre una matriz numérica que representa digitalmente a una imagen.

1.7.3. Formulación Local Algorítmica

Cuando hemos descrito el proceso de convolución como un método básico en el tratamiento digital de señales, hemos dado por supuesto que las operaciones que incluía eran sumas y productos. Sin embargo, si el método es general, deberá ser independiente de la selección de operadores. Así, podemos proponer una formulación más general, a la que llamamos convolución algorítmica, de la siguiente manera:

1. Organizar el espacio de datos de entrada en forma de memoria FIFO, que apila un cierto número de imágenes de entrada.
2. Organizar el espacio de salidas en forma de memoria FIFO, que apila las respuestas durante un cierto número de intervalos de tiempo de cada uno de los procesadores locales que actúan sobre la imagen.
3. Formular la acción de filtrado espacio-temporal en términos del proceso local que realizan una serie de módulos de cálculo que toman datos de ambos espacios, de acuerdo con dos núcleos y los procesan en paralelo, cargando el espacio de salidas con el resultado de su proceso local. Después se actualizan las FIFO de entrada y salida.
4. Descomponer las zonas de toma de datos en dos regiones: (**centro** y **periferia**), especificando su forma o volumen (esferas concéntricas, elipsoides de revolución, etc….
5. Especificar las operaciones que se realizan sobre el centro, las que se realizan sobre la periferia y las que combinan los resultados de ambas.
6. Usar reglas ("*si* **condición** *entonces* **acción**") para describir todos los procesos locales. Si los campos de condición incluyen operadores lógico-relacionales, tenemos filtros algorítmicos

Obsérvese que en toda la descripción anterior no hemos hecho ninguna referencia explícita al tipo de operaciones que se realizan sobre la parte central y la periferia de el campo de datos sobre el que se está calculando. Si las operaciones que se realizan son productos y sumas, tenemos la formulación anterior como caso particular. Sin embargo, ahora podemos formular cualesquiera otros procesos, por ejemplo: Buscar el máximo en el centro, buscar el máximo en la periferia y compararlos. Si el mayor pertenece al centro,

darlo como respuesta. Si pertenece a la periferia, traerlo an centro y darlo como respuesta. En este caso, el filtro actúa como un ***proceso cooperativo*** que detecta localmente y propaga los valores máximos en una región de la imagen. Veremos también que esta formulación permite umbralizar de forma adaptiva y realizar otras muchas tareas. Finalmente, veremos como es también útil para diseñar redes de neuronas artificiales.

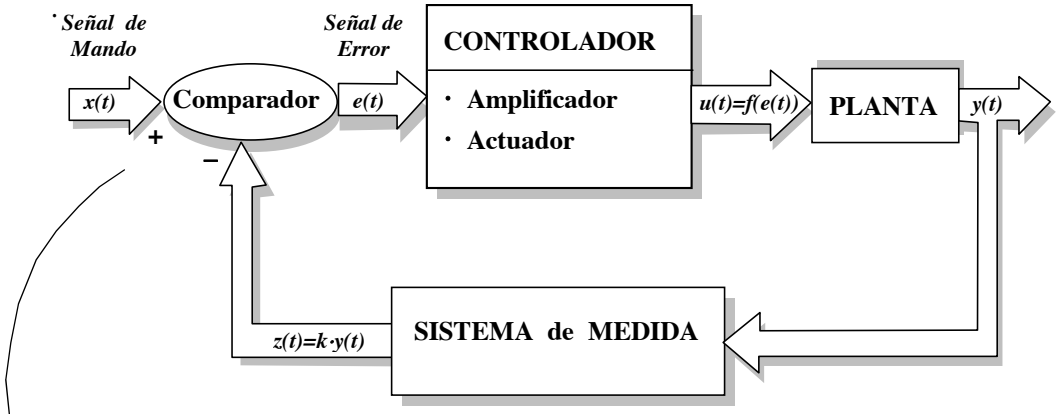
1.8. Las Tareas de Supervisión y Control a Nivel de Conocimiento

La primera parte del temario está dedicada a la "***percepción***", entendido el término en un sentido bastante limitado. De hecho, tras mencionar otros tipos de sensores usuales en robótica, dedicaremos la mayor parte del esfuerzo a la visión de bajo nivel y a los aspectos básicos de la visión de alto nivel (reconocimiento de objetos) en situaciones estáticas, sin considerar los procesos asociados al reconocimiento de objetos móviles. Aquí, partimos siempre del medio externo y buscamos un modelo que represente a ese medio.

La segunda parte del temario, aborda el ***problema complementario***: partiendo de una acción que queremos ejecutar, monitorizar el proceso de ejecución y controlar su ejecución a través de un esquema de realimentación negativa. Para modelar esta tarea a nivel de conocimiento, podemos partir del diagrama de bloques de la figura 15. El propósito de todo sistema de control es mantener constante el valor de una magnitud, *y(t)*, o hacer que esa magnitud siga la evolución temporal de una señal de mando, *x(t)*. Para conseguirlo se parte de un análisis del problema a nivel estratégico y se identifican las funciones que se consideran necesarias (**comparación**, **amplificación**, **actuación** y **medida**), y su estructura de conexión: lazo de realimentación que compara el valor de la magnitud de salida *y(t)*, con el valor de consigna, *x(t),* y actúa para minimizar la diferencia que se está midiendo constantemente para informar al comparador. Obsérvese que este análisis y el esquema resultante son totalmente independientes de si el control es analógico o digital y de cuál sea la función de transferencia de la planta. En cada caso, los ***métodos***, las formas de ***representar*** el conocimiento y los mecanismos de ***inferencia*** variarán, pero la organización de la tarea permanece. Su esquema conceptual es el mismo.

Así, inicialmente, la tarea de control pueden descomponerse usando distintos métodos. Por ejemplo, para la comparación podemos usar la resta aritmética, un algoritmo digital de resta o un método de medida de semejanzas entre patrones que establece la analogía a un nivel alto y después refina. Análogamente la representación del conocimiento es la adecuada a los tres métodos: variables analógicas, vectores booleanos o marcos y redes. Finalmente, la inferencia es la correspondiente a estas representaciones: analítica en el caso analógico, lógico-deductiva en el digital y búsqueda heurística en el caso de las matrices de semejanza. Lo relevante a nivel de estructura de tareas es que en todos los casos estamos realizando la misma subtarea (***comparación***).

Las situaciones reales, en general, no son tan sencillas como la propuesta en este esquema básico. En algunos casos, antes de plantear el problema concreto de control y/o supervisión de un proceso, es necesario realizar un estudio amplio sobre el sistema que se quiere controlar o sobre el sistema de control que se quiere supervisar para identificar las **dependencias causales** entre las distintas variables de las que se conocen sus valores porque son medibles en puntos específicos de un sistema. Estas dependencias causales junto con el conocimiento global del proceso constituyen la descripción en lenguaje natural de la tarea.



SUBTAREA	Método	Representación del Conocimiento	Inferencia
Comparación	◆ Resta aritmética e = x - k·y	◆ Variables Analógicas	◆ Analítica
	◆ Algoritmo Digital	◆ Vectores Booleanos ◆ Lógica	◆ Deductiva (Reglas)
	◆ Matrices de Semejanza (Establece y refina)	◆ Marcos ◆ Redes ◆ Reglas	◆ Búsqueda Heurística

Fig. 15. Diagrama de bloques de un sistema de control usado para ilustrar su modelado a nivel de conocimiento. (Tomada de J. Mira y A.E. Delgado, fig.2.8, cap. 2 Libro de *"Introducción a la IA"*).

De esta descripción en lenguaje natural arranca el modelo conceptual a nivel de conocimiento. Además, las relaciones inferenciales y los conceptos que se usan pata modelar el conocimiento del dominio **son muy dependientes de la aplicación**. Tres ejemplos distintos podrían ser:

(1) Identificación y supervisión de los circuitos de control en el lazo de "agua de alimentación" de una central nuclear.

(2) Supervisión y vigilancia del control del proceso de producción de azúcar a partir de remolacha.

(3) Control de la navegación de un robot autónomo en un medio estructurado.

En nuestra asignatura haremos referencia a la supervisión "inteligente" del control de procesos, pero nos centraremos más en los aspectos básicos de la **robótica**. Esto nos permitirá enlazar la primera parte (que terminaba con un modelo del medio) con la segunda parte, dedicada a la navegación en ese medio del robot que ha construido el modelo. Así, la integración entre **percepción** y **acción** será relativamente evidente.

1.9. Integración Percepción-Acción

Una de las tareas básicas de la IA en su intento de modelar el conocimiento humano no analítico es tratar de integrar tareas perceptivas con tareas motoras de una forma no trivial. Los seres vivos actuamos para percibir("andamos y movemos la cabeza para ver") y percibimos para actuar ("miramos para caminar"), de tal forma que no está claro donde empieza y donde acaba el lazo de percepción-acción.

En robótica autónoma hace tiempo que se ha reconocido que muchos problemas perceptuales, relacionados con la construcción de un modelo del medio, no son fáciles de comprender si no se considera claramente que este **sistema sensorial** está **integrado** en un modelo conceptual más amplio que tiene en cuenta que el robot tiene que moverse navegando en ese medio, con la descripción limitada que le proporcionan sus sensores etc… Inversamente, no tiene sentido buscar precisión en una modalidad sensorial si su resultado no va a ser necesario para guiar las tareas de navegación que se están considerando.

Como la visión artificial es muy pobre en comparación con la visión humana, parece razonable empezar a pensar en las tareas de acción de forma limitada, esquemática y protocolaria, para que la riqueza de la percepción sea análoga a la de la acción y la primera forma de **integración** (**establecer correspondencias** entre configuraciones del medio y configuraciones de acción) sea fácil de alcanzar. Probablemente, una perspectiva análoga a esta, junto con el intento de imitar a la biología fue la que llevó al grupo del Prof. Warren S. McCulloch a estructurar las estrategias de control de la acción en términos de un conjunto de "**modos de comportamiento**", en general pocos y mutuamente exclusivos. Si el sistema está en un modo, no puede estar en ningún otro a la vez. La arquitectura de integración entre percepción y acción que se inspira en esta forma de relacionarse las configuraciones sensoriales con las decisiones estratégicas sobre "qué hacer ahora", tiene la ventaja de ser computable sin ser trivial [14,15].

Podemos describirla usando el modelo conceptual de integración que se muestra en la figura 15. Tras el preproceso y la integración plurisensorial obtenemos una representación del medio que enlaza, por una parte, con las tareas de **decisión** y **planificación** y por otra con la etapa de salida previa al control de efectores (escenarios de bajo y alto nivel y generadores de planes de alto y bajo nivel). Esta última conexión permite comportamiento **reflejo** y **navegación reactiva**. Es decir, ante determinadas circunstancias se inhibe todo tipo de proceso elaborado y se **reacciona** rápidamente, de acuerdo con patrones que están almacenados en **tablas** o en autómatas finitos con muy pocos estados.

El módulo de **decisión** y **control** está comunicado bidireccionalmente con la memoria que almacena el grafo con el que representamos al medio, incluyendo la posición del robot en ese medio ("propiocepción") y su función es considerar el estado global del sistema y seleccionar un **modo** de acción y una **forma** de percepción. En su implementación, puede ser un SBC convencional o cooperativo, con arquitectura modular donde cada módulo defiende "**su modo**" y, en cada situación domina quien posee más información relevante

para esa situación. Así, se garantiza **redundancia** en la función de mando y cierta tolerancia a fallos. Si algún módulo decisor deja de funcionar, el resto asume su función.

Fig. 15. Estructura general de una arquitectura para la integración de percepción y acción en el sentido de McCulloch. La percepción termina en un grafo que modela el medio y la acción se organiza en términos de un conjuto pequeño de modos incompatibles, Entre ambos hay un proceso bidireccional de decisión que enlaza el modelo con los modos. Obsérvese que los datos sensoriales actúan en la vía directa y sobre los modos. Análogamente, la representación puede actuar de forma "refleja" sobre el generador de planes.

Siguiendo con la figura 15, vemos que el siguiente paso es la selección de uno de esos modos. En su origen, la formulación del problema pretendía modelar el comportamiento de una estructura nerviosa en las vertebrados (la formación reticular) encargada de decidir esos modos que en biología son del tipo: comer, defenderse, atacar, huir, buscar relación sexual, etc… En robótica son cosas menos poéticas y hablamos de modos para referirnos a segmentos protocolarios de comportamiento tales como: avanzar, retroceder, buscar ciertas configuraciones ("puertas",…), alinearse, coger objetos, etc… y todos esos otros que consideremos necesarios a la hora de construir el modelo para una aplicación específica. Los modos son los componentes "standard" y reutilizables de los que disponemos para sintetizar cualquier segmento de conducta en un robot.

Una vez que se ha decidido un **modo** se comunica en los dos sentidos. Por un lado se comunica al generador de planes y al control de efectores, para que se lleve a cabo. Por otro lado, se comunica al módulo sensorial para que se adapte ("sintonice") en su representación del medio a la forma más conveniente para ese modo (¿a qué hay que prestar atención?). Finalmente, la información del modo seleccionado también se manda al modelo del medio para que se adapte a la configuración más adecuada a ese modo. Cada vez que se cambia de modo, se actualiza el modelo del medio.

La etapa final, dentro de un modo y antes de llegar a los efectores es la **planificación estratégica**. El modo decidido entra a un generador de planes que también considera la entrada sensorial y los escenarios asociados a ese modo.

El problema de la planificación espacial y temporal en IA tiene mucha entidad y no es necesario ni conveniente extendernos aquí en su exposición. Sólo la hemos mencionado por completar el panorama de la integración entre percepción y acción.

Finalmente queremos resaltar para nuestros alumnos que lo importante no es el uso de esta u otra arquitectura para integrar, sino el hecho de que alguna arquitectura que integre percepción y acción siempre será necesaria y que, en mayor o menor medida, tendrá que considerar todos o algunos de los elementos que hemos mencionado aquí.

REFERENCIAS

[1] Mira, J. et al. (1995): Cooperative Processes at Symbolic Level in Cerebral Dynamics: Reliability and Fault Tolerance. In Moreno R. and Mira, J. (eds.): *Brain Processes, Theories and Models*. The MIT Press. pp. 244-255

[2] Mira, J. (1998): Operaciones "Inteligentes" en Sistemas Artificiales: La Perspectiva de la Inteligencia Artificial en la Comprensión del Sistema Nervioso. *Revista de la Real Academia de Medicina de Catalunya*. Vol 12. Sup. 1. pp. 87-107.

[3] Mira, J. (1998): Inteligencia Artificial, Emoción y Neurociencia. (Pendiente de publicar Revista *Arbor*).

[4] Mira, J., Delgado, A.E., Boticario, J.G., Díez F.J. (1995): *Aspectos Básicos de la Inteligencia Artificial*. Sanz y Torres, S.L. Madrid,

[5] Marr, D. (1982): *Vision*. Freeman, New York.

[6] Newell, A. (1981): The Knowledge Level. *AI Magazine*, pp. 1-20.

[7] Maturana, H.R. (1975): The Organization of the Living: A theory of the Living Organization. *Int. J. Man-Machine Studies*, 7, pp.313-332.

[8] Varela, F.J. (1979): *Principles of Biological Autonomy*. North-Holland. New York.

[9] Mira, J., Delgado, A.E. (1987): Some Comments on the Antropocentric Viewpoint in the Neurocybernetic Methodology. Proc. of the Seventh International Congress of Cybernetics and Systems, Vol. 2. London. pp. 891-895

[10] Mira, J., Delgado, A.E. (1995): Computación Neuronal Avanzada: Fundamentos Biológicos y Aspectos Metodológicos. En Barro, S.; Mira, J. (eds.): *Computación Neuronal*. Cap. VI, Servicio de Pub. e Intercambio Científico. Univ. de Santiago de Comp.. pp. 125-178.

[11] Mira, J., Delgado, A.E. (1997): Some Reflections on the Relationships Between Neuroscience and Computation. In Mira, J.; Moreno-Díaz, R.; Cabestany, J. (eds.): *Biological and Artificial Computation: From Neuroscience to Technology*. LNCS, 1240. Springer-Verlag, Berlin. pp. 15-26

[12] Mira, J. (1997): Modelar Conocimiento como Tarea Básica en Inteligencia Artificial y en Redes Neuronales Artificiales. En el curso de verano: *Aspectos Básicos de la Inteligencia Artificial*. pp I.1-I.24. UNED.

[13] Mira, J., Herrero, J.C., Delgado, A.E. (1998): "Where is Knowledge in Computional Intelligence?: On the Reduction of the Knowledge Level to the Level Below". *Proceedings of the 24th EUROMICRO Conference*, *IEEE,* Vol. II, 723-732.

[14] Moreno-Díaz R. and Mira, J. (1987).: "Architectures for Integrating Artificial Perception and Action". *Cibernetica, Ordinadors I Teoria de Sistemes*. *Interkibernetik'87. 205-211*.. J.C. Palavecino, ed. Tarragona

[15] Romo, J. de la Paz, F. Mira, J. (1998): "Incremental Building of a Model of Environment in the Context of the McCulloch-craik's Functional Architecture for Mobile Robots". *Methodology nad Tools in Knowledge-Based Systems*. J. Mira, A.P. del Pobil and Moonis Ali, eds. LNAI, 1415. 337-352. Springer-Verlag, Berlin,