VNIVERSIDAD NACIONAL AVTÓNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"GENERACIÓN AUTOMÁTICA DE REGLAS PARA LA MÁQUINA DE INFERENCIAS CLIPS"

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS (COMPUTACIÓN)

P R E S E N T A:

VERÓNICA ESTHER ARRIOLA RÍOS

DIRECTOR DE TESIS:

Dr. Jesús Savage Carmona

México, D.F. 2006.

RESUMEN

En este trabajo se procuró establecer un esquema para pasar de la percepción de información visual en forma simbólica, a la adquisición de reglas de comportamiento entre los objetos identificados. Se utilizó como guía un micromundo en el cual es posible distinguir algunas formas básicas (manos, pies, calcetines, zapatos) que se juntan y separan. Es importante aclarar que el sistema desconoce cualquier información acerca de las funciones de los objetos que percibe o la forma en que éstos se relacionan.

El proceso se dividió en tres etapas:

- Recepción de información visual simbólica.

Consiste en la recepción de información sobre objetos observados, así como sus características distintivas. Se utilizaron los conceptos clave de orientación a objetos (OO) para diseñar un modelo de representación interna donde fuera posible establecer relaciones entre los objetos a través de la jerarquía de herencia. Dicha jerarquía es extendida dinámicamente conforme el agente detecta objetos nuevos e interacciones espaciales entre ellos.

- Generación de la representación interna.

Se establece una clara diferenciación entre la información que percibe el agente desde el exterior a través de su sistema de visión y la representación que da internamente a dicha información, así como la forma en que la manipula.

En esta etapa, se hace hincapié en la diferencia entre el concepto de clase e instancia (tomados de OO) y se utilizan para diferenciar entre la descripción memorizada del objeto (que en este trabajo corresponde a la clase) y el objeto concreto que se "está viendo" (el cual es representado aquí con una instancia). Una vez hecha esta distinción, es posible realizar el seguimiento de los movimientos realizados por las instancias en observación.

- Extracción reglas.

Las modificaciones observadas en los estados de los objetos participantes son escritas por el sistema en forma de funciones. (Concretamente como funciones ejecutables del lenguaje de programación LISP)

Finalmente se realiza internamente un análisis de la secuencia de acciones observadas, para determinar los órdenes parciales en que pueden ser realizadas. Extraída esta información, es posible reescribirla como reglas para una máquina de inferencias (CLIPS, en este caso).

Dedicatoria

A Dios, origen de todo

A mis padres

A mi hermano

A mi tutor del posgrado, el Dr. Jesús Savage, por apoyarme en mis más descabelladas aventuras (esta tesis incluida).

A mis profesores: Elisa Viso, Ana Luisa Solís, Salvador López y Sofía Natalia, por enseñarme el camino.

A Isaac Orihuela, Miguel Ángel Téllez, Abraham Valle, mis compañeros de "Macroware" y tantos amigos más con quienes compartí mis mejores años en la facultad de ciencias.

A Olga, Octavio y los demás chicos del posgrado: ¡tantas gracias!

A Martin Murray y Nick Mancuso, por estar a mi lado en una etapa muy especial de mi vida.

Al apasionado por la inteligencia artifical.

A todas aquellas personas que no he podido citar porque nunca acabaría.

AGRADECIMIENTOS

Deseo agradecer especialmente:

A mi director de tesis, el Dr. Jesús Savage.

A mis sinodales: Dr. Boris Escalante, Mat. Ana Luisa Solís, Dra. Ana Lilia Laureano y Dra. Sofía Natalia.

Al CONACYT por su apoyo económico.

Al IIMAS y a todas la escuelas e institutos vinculados al Posgrado en Ciencia e Ingeniería de la Computación.

ÍNDICE

CAPÍTULO I

INTRODUCCIÓN

1.1 MOTIVACIÓN Y PROBLEMÁTICA

Los planteamientos de los años 60, cuando apenas surgía la inteligencia artificial, se caracterizaron por ser ambiciosos. Uno de sus principales íconos fue la prueba de Turing. Esta prueba exige programar a una computadora de tal modo que sea capaz de hacerse pasar por un ser humano. En un principio, se propuso que esta prueba fuese realizada al nivel del lenguaje escrito, posteriormente, investigadores más ambiciosos incluyeron otros niveles de comunicación, como lenguaje oral o incluso la fabricación de robots humanoides.

Emular las capacidades de los seres humanos para resolver problemas, producir e interpretar el lenguaje e incluso aprender de la experiencia, resultó ser una tarea mucho más compleja de lo que se esperaba en un principio. Los especialistas se encontraron con barreras que no serían superadas con facilidad.

Por ejemplo: programas que alcanzaban resultados sorprendentes en dominios restringidos, resultaron utilizar métodos imprácticos para medios más realistas. Sistemas que pretendían conversar con personas, fallaban rápidamente fuera de contextos rígidamente delimitados. La presencia constante de ruido y errores en la información recibida, obligó al surgimiento de técnicas que pudieran extraer resultados útiles tomando en cuenta los contratiempos citados. La evolución de la inteligencia artificial tuvo que detenerse en cada uno de sus pasos, para desarrollar las herramientas necesarias para atender y formalizar la problemática descubierta.

Para los años subsiguientes, la naciente disciplina se enfocó en el estudio de problemas muy particulares y limitados, que podían resolverse depurando técnicas de búsqueda sobre una base de conocimientos. Se observó que el problema de representar el conocimiento requerido es altamente complejo por sí mismo. Incluso, a la fecha, no se ha encontrado una solución enteramente satisfactoria. Enfoques como el uso de redes neuronales o algoritmos genéticos desaparecieron casi por completo de la escena por varios años, otorgándose preferencia a las representaciones simbólicas. Gran parte del conocimiento se le entregaba a la máquina, introducido directamente por el programador. La visión de "reproducir" a un ente inteligente completo, se perdió.

Sin embargo, tras el desarrollo a conciencia de los elementos requeridos y la evolución de los sistemas de robótica, esta situación está cambiando. La aparición y posterior popularización del concepto de "agentes racionales",

devuelve la vista hacia el objetivo original. Plantear y analizar un problema de IA en términos de agentes, involucra considerar entradas a través de "sensores" y salidas mediante "actuadores"; retoma la visión de crear un ente racional y facilita el volver la visión, una vez más, hacia una comparación del agente artificial con los seres vivos que habitan este planea.

Figura 1.1 Un robot que percibe su medio a través de sensores, analiza la información y actúa mediante actuadores es capaz de realizar una actividad compleja como manejar una bicicleta [1].

No es que los problemas hayan cambiado, o que su dificultad haya sido disminuida, o que la importancia de las búsquedas y las formas de representar el conocimiento haya disminuido. El cambio importante se encuentra en la manera de enfocar estos problemas. Es con este enfoque, que la presente tesis pretende explorar los problemas que se encontrará el investigador al intentar integrar diversos módulos de percepción y acción, tanto de agentes virtuales como de robots, en un ente capaz de percibir su entorno y analizarlo a partir de sus percepciones. Este agente deberá crear su propia representación interna y extraer reglas de comportamiento. Su objetivo será adquirir un lenguaje simbólico que le permita, posteriormente, alcanzar un cierto "conocimiento conciente" de su medio, de su contexto.

Para alcanzar este objetivo se analiza el proceso de extracción de reglas de comportamiento, a partir de la experiencia, para un problema concreto. Estas reglas podrán ser utilizadas por una máquina de inferencias para realizar planeación no lineal. A lo largo de este análisis emergen detalles

importantes que deben ser considerados para la resolución general del problema planteado.

1.2 Objetivos Principales

El objetivo primordial de esta tesis es estudiar la extracción de reglas para un planificador de acciones a partir de ejemplos concretos que percibe un agente.

La meta es completar un programa que sea capaz de analizar una o varias secuencias de acciones mediante las cuales se realiza una actividad concreta. A partir de este análisis, el sistema debe extraer las reglas esenciales que definen la acción. Cada secuencia es percibida por el agente como una sucesión de descripciones de imágenes previamente catalogadas, pero sin mayor conocimiento ontológico.

1.3 Contribución y Relevancia

En el estado actual de la inteligencia artificial y la robótica, el problema de integrar el ciclo percepción-análisis-acción se encuentra abierto para agentes racionales complejos. Es decir, agentes capaces de utilizar el sentido común e interactuar vía lenguaje natural con los seres humanos.

Dos problemas que resultan centrales para poder atacar este problema son:

- La generación dinámica de la representación interna del mundo a partir de la información recibida por la vía sensorial y
- La extracción de reglas de comportamiento a partir de las experiencias percibidas.

Aunque en este trabajo sólo se inicia el tratamiento de estos problemas para un caso concreto, este estudio provee de una visión comprensible del sistema completo y de la problemática que plantea ya como un todo y no sólo como módulos separados. Esta unificación, así mismo, se muestra como el camino para resolver problemas que requieren de manejo de contexto, como los mencionados anteriormente. A lo largo de este trabajo se vislumbran problemas de decisión y aprendizaje que son prerrequisitos indispensables para un agente inteligente. Consciente e inconscientemente, estos problemas han sido asignados a los seres humanos, en lugar de permitir que la computadora realice la elección. Esta asignación se debe, fundamentalmente, a que dotar a una máquina de esa habilidad no es trivial. Sin embargo no se puede eludir el problema por siempre.

Una primera aproximación histórica a este problema es el programa SHRDLU, realizado por Winograd en 1970 [2]. SHRDLU es el primer representante de los programas que intentan utilizar el conocimiento acerca de su entorno para resolver problemas.

El ambiente virtual incluye una simulación de las leyes de la física que rigen al mundo de los bloques. El agente SHRDLU conoce algunas acciones básicas como "tomar un bloque", "colocar un bloque sobre...", etc. y es capaz de experimentar con ellas. Especialmente, Winograd trata la desambiguación en lenguaje natural, aprendizaje de vocabulario y métodos nuevos, detección de acciones que llevan a resultados erróneos (como colocar un cubo sobre una pirámide) e incorpora el conocimiento adquirido a través de la experiencia en su base de conocimientos.

Sin embargo SHRDLU está limitado al mundo de bloques y al idioma inglés. Esto es porque su propio aprendizaje de lenguaje, fue programado para funcionar con la estructura gramatical del inglés. Muchos trabajos que se realizan en la actualidad, también se encuentran restringidos al idioma inglés, pues el conocimiento se les entrega, tras una preparación previa, en ese idioma; no es adquirido directamente por el sistema.

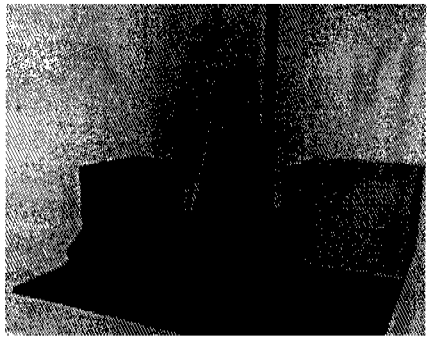


Figura 1.2 Mundo de bloques de SHRDLU. El ambiente virtual incluía una simulación de las leyes físicas que afectan el comportamiento de los bloques [3].

En este trabajo no se pretende siquiera acercarse a las capacidades de SHRDLU, que es el programa más completo, pero sí se retoma su filosofía, la cual expresa Winograd en la frase siguiente:

> *"Una computadora no puede manejar razonablemente el lenguaje a menos que pueda comprender la materia sobre la cual habla"* [2]

En esta tesis se ahonda en el punto de "comprender la materia". Se pone especial énfasis en la restricción de que: toda la información nueva recibida, debe provenir exclusivamente de los sensores del agente.

Adicionalmente, se comienza a trabajar con la reducción de secuencias complejas de acciones elementales, a símbolos internos que se refieren a un conjunto de ellas. La incorporación de nuevas rutinas ya había sido incorporada en SHRDLU; sin embargo, en este trabajo será el propio agente el que se encargue de determinar qué secuencias pueden ir siendo agrupadas dentro de una sola rutina. El agente es capaz de representar secuencias complejas como sucesiones de acciones con un menor grado de granularidad. Estos subcomponentes pueden ser abstraídos por ella misma y descompuestos en sus partes, en caso de ser requerido. Sin este paso, no resulta evidente como enfrentar el problema de detectar qué acciones deben

ser realizadas en estricto orden y cuáles no; las razones serán analizadas en el desarrollo de esta tesis.

1.3 Vista General de la Tesis

En el capítulo 2 se exponen los antecedentes históricos y algunas discusiones relacionadas con la inteligencia artificial y el aprendizaje en agentes virtuales. Así mismo, se describe en detalle el problema que guía esta investigación y la motivación para haberlo elegido. En el capítulo 3 se describe el funcionamiento de un planificador, el apéndice A detalla el funcionamiento de la máquina de inferencias CLIPS, para la cual se generan las reglas aprendidas en la implementación concreta de los algoritmos propuestos.

Los capítulos siguientes contienen la propuesta de esta tesis. Las estructuras y algoritmos expuestos tienen su origen enteramente en este trabajo; salvo por algunas ideas básicas tomadas del paradigma de programación orientada a objetos y técnicas relacionadas a la programación funcional, que son explicadas someramente antes de ser utilizadas.

En el capítulo 4 se expone la representación del mundo. Esta representación surge inspirada por el uso de marcos (de donde emerge la programación orientada a objetos) pero con características muy propias. Dicha representación fue significativamente simplificada para hacer viables los análisis que deben hacerse de la información. Dada su alta complejidad, el caso general no podría haber sido cubierto, de ningún modo, por este trabajo. Aquí se detalla la forma en que el agente recibe la información a través de un sistema de visión simulado y cómo asigna una identidad a cada tipo de imagen percibida.

En el capítulo 5 se expone cómo el agente representa internamente el estado del mundo y sus transformaciones. El agente asocia una identidad a cada objeto que ve e interpreta cambios entre imágenes consecutivas como movimientos de estos objetos.

El capítulo 6 explica cómo el agente analiza la secuencia de acciones observada y extrae la mayor cantidad de información posible de ellas. Tras este análisis, la información requerida para redactar reglas para la máquina de inferencias ya ha sido recopilada. Por último se realiza la traducción a la sintaxis requerida por CLIPS.

A lo largo de cada capítulo se van presentando los resultados obtenidos, para la fase correspondiente, por la implementación de las ideas propuestas. Esto es con la finalidad de establecer claramente las bases para los capítulos subsiguientes.

Finalmente en los capítulos 7 y 8 se analiza el proceso completo y los resultados obtenidos. Se indica también, la ruta para continuar con esta investigación.

CAPÍTULO II

ANTECEDENTES

2.1 EL OBJETO DE ESTUDIO DE LA IA: EL SER HUMANO

Desde Aristóteles hasta la fecha, el ser humano ha hecho numerosos intentos por encontrar reglas que describan el funcionamiento de su mente. Actualmente, gracias a la invención y construcción de las computadoras, estos intentos se han llevado al límite de proponerse el reto de implementar estas leyes en un sistema físico que no sea el cerebro humano. Ésta es, para muchos, la meta final de la Inteligencia Artificial.

Por lo expuesto anteriormente, se inicia el presente trabajo con un rápido repaso de todos aquellos elementos involucrados en el comportamiento humano. No es posible pretender ser exhaustivos, pero se incluye esta síntesis para luego considerar la estructura del sistema a diseñar.

En principio, es fundamental considerar que este mundo tiene 4 dimensiones: tres espaciales y una temporal. Para poder ubicar aquello que hay y sucede se requiere, por lo tanto, de un sistema de representación interna que ubique las percepciones del individuo en estas 4 dimensiones.

Para ayudarle a comprender aquello que le rodea, el ser humano cuenta con diversos sentidos, que le brindan diferentes tipos de información acerca de los objetos en su medio ambiente: su posición en el espacio, composición física y química, así como los cambios que sufren a través del tiempo.

Cada individuo debe saber elegir entre aquello que le puede ser de interés y aquello que no le es de utilidad en el momento, para saber a qué otorgar una representación interna y abstracta en su memoria y a qué no (aquello que es digno de ser olvidado), de modo que sea capaz de recordar lo que es útil y razonar con ello.

Claramente, la función de los seres humanos no se reduce a la de ser sólo espectadores, cada uno tiene la capacidad de interaccionar con el mundo que le rodea, influir en él, modificarlo. Ser parte de él y de su evolución.

Como todo ser vivo, las personas tienen necesidades fisiológicas, requieren de alimentos para generar las células que las constituyen y la energía que las mueve; de una cierta temperatura para continuar funcionando, del sueño, etc.

La supervivencia de cada individuo depende de su capacidad para interpretar la información que recibe, interrelacionarla y generar comportamientos acordes a cada circunstancia, buscando siempre la actitud más provechosa dependiendo de la ocasión: aquella que le ayude a satisfacer sus necesidades.

Además de capacidades receptoras, cada persona cuenta con habilidades motoras que le habilitan para transformar aquello que le rodea. Más aún, cuenta con la capacidad de percibir sus propias acciones, movimientos, contactos con el exterior y recibir retroalimentación. Esto, gracias a que recibe información sensorial acerca de su propio estado con respecto a sus alrededores y de su movimiento. Siente la fuerza que se requiere para caminar, para empujar un objeto o el movimiento de sus mandíbulas al masticar los alimentos, los movimientos de su boca y laringe al pronunciar sonidos, entre muchas otras cosas.

A cada instante, el individuo puede corroborar si su representación interna del mundo es congruente con aquello que percibe a través de los sentidos. Esto no sólo ocurre en el sentido físico (ubicación espacial, peso, etc.) si no también en el emocional e intelectual: Puede hacer uso de la empatía, adivinar lo que otros sienten ante determinadas circunstancias y comprobar si sus expectativas con respecto a las acciones y pensamientos de los demás concuerdan con sus experiencias. Es capaz de comprobar si sus percepciones del mundo corresponden a lo que ha predicho dentro de si.

Sin embargo, trabajar con la información simplemente a nivel sensorial carece de sentido para cualquier individuo. Las personas no acostumbran pensar en términos de señales. Más allá de las sensaciones intuitivas (de las cuales se debe desconfiar en ocasiones), las señales son interpretadas y se piensa en términos de símbolos que se refieren a un conjunto de ellas. Los símbolos son abstracciones de percepciones sensoriales y su ubicación entre los pensamientos es más bien puntual que difusa. La existencia de esta representación simbólica dentro de una región del cerebro humano es la que permite el desarrollo del lenguaje y el pensamiento lógico [5].

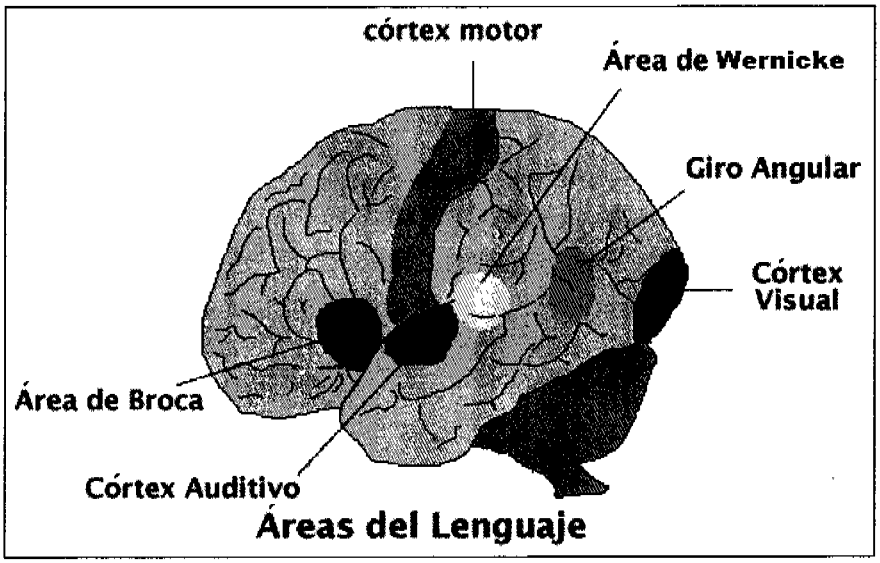


Figura 2.1 Regiones físicas del cerebro involucradas en el manejo del lenguaje. El área de Broca se especializa en el habla, mientras que el área de Wernicke en la compresión del lenguaje. El giro angular conecta a los centros del lenguaje con el córtex visual, permitiendo la lectura y escritura del lenguaje. Las palabras habladas son tratadas como símbolos, mientras que, al parecer, las palabras escritas son tratadas por estas regiones como objetos [6].

Además, el ser humano no es un ente solitario dentro de una coraza a la cual llega toda la información para ser analizada. Cada persona es parte de una sociedad y los individuos pueden comunicarse entre ellos e intercambiar expresiones e impresiones.

Si bien es cierto que cada uno tiene su propia representación interna del exterior, varios individuos pueden estar conectados por fenómenos externos comunes. Pueden encontrarse en presencia de las mismas vibraciones (sonoras o electromagnéticas) o experimentar la acción de las mismas fuerzas. Esto originará una relación directa entre las representaciones internas de cada uno de ellos, pues todos deberán asociar algún símbolo a ese fenómeno que presencien en común.

Aún cuando la relación entre las representaciones internas de cada individuo no sea uno a uno, en presencia de fenómenos comunes siempre deberá haber algo común y este punto compartido les proveerá de una vía para comunicarse [7].

En particular, gracias al establecimiento de consensos es posible asociar rasgos concretos a un símbolo común. Este hecho es el fundamento de la comunicación a través del lenguaje.

Ya que los seres humanos pueden comunicar y recibir aquello que han generado dentro de sus mentes (ya sea a través de gemidos, señas, lenguaje oral o corporal), ya no sólo pueden influir en su ambiente en una forma mecánica, sino también en las mentes de sus semejantes.

De este modo, la vida de un ser humano transcurre entre percepciones sensoriales ligadas a su realidad física y en el constante intercambio de imágenes mentales con sus semejantes, a través de la simbología correspondiente.

Este intercambio es fundamental para la evolución de la especie, pues la experiencia propia y el azar genético dejan de ser las únicas fuentes de adquisición de respuestas a problemas. Gracias al lenguaje (en cualquiera de sus manifestaciones) la humanidad evoluciona como sociedad y no sólo como individuos. Las lecciones aprendidas por otros pueden ser compartidas con sus semejantes, ahorrándoles la ardua tarea de experimentarlo todo de nuevo por sí mismos y permitiéndoles continuar donde los anteriores se quedaron. En realidad, el deseo de comunicarse con sus semejantes es un instinto inherente al ser humano y una necesidad para la evolución de la sociedad.

Por último, el ser humano no solamente puede razonar acerca de su mundo exterior, también puede hacerlo sobre sus propios sentimientos y razonamientos. Esta capacidad de introspección le permite modificar su comportamiento a voluntad y evolucionar conscientemente su proceso mismo de aprendizaje.

Los métodos de razonamiento pueden ser diversos, las formas de atacar diferentes problemas no son únicas, pero para las personas es posible aprenderlas y elegir las más adecuadas en cada caso. A veces se trata de

experimento y error, en ocasiones de aplicar algoritmos predefinidos, otras veces se hace uso de deducción o inducción; lo que cabe destacar es que: Un solo individuo no se encuentra limitado a un solo método o comportamiento predefinido; aún cuando ciertos rasgos característicos de su personalidad le lleven a inclinarse más por unas decisiones que por otras, entre mayor sea el conocimiento que posee de sí mismo, mayor es el control que posee sobre sus propios actos y pensamientos y más son las opciones que tiene a su disposición en cada momento de su vida.

En realidad, la Inteligencia Artificial (IA) es la última manifestación del deseo nato del ser humano por comprenderse y emularse a sí mismo, en términos de los símbolos que ha creado dentro de sí.

Tal vez esta disciplina haya surgido hace relativamente muy pocos años, pero el intento de los seres humanos por mecanizar aquello que ya han logrado resolver una vez, es un instinto nato. Por ejemplo, una vez que un niño ha aprendido a caminar, ya no vuelve a preocuparse por cómo hacerlo; una vez que ha aprendido a multiplicar, ya no necesita recordar el origen de las tablas y el algoritmo, simplemente los usa.

La invención de las computadoras obedece al deseo de evitar las labores repetitivas (y por lo tanto tediosas) y a la necesidad de mecanizar algoritmos que son muy complejos (o largos) como para almacenarlos en la memoria propia (en el cerebro). En lugar de ello, la información y los procedimientos predefinidos son almacenados en algún dispositivo externo y se hace que una máquina los ejecute. De este modo los recursos personales quedan libres para que el individuo se ocupe, mejor, de aquello que le resulte novedoso y, por lo tanto, de mayor interés. Un problema, una vez resuelto, pierde su magia. Hasta ahora, el trabajo de encontrar dichos algoritmos permanece reservado a las personas.

Ya que, el ser humano ha adquirido algún entendimiento de su propio mecanismo de razonamiento y sus procesos de aprendizaje, lo natural es querer enfrentar un nuevo reto, dar el siguiente paso: que las máquinas que él construye puedan hacerlo también automáticamente, que fueran capaces de aprender y generar algoritmos. Este es el más grande desafío de la Inteligencia Artificial.

Sin embargo, para algunos, tampoco basta con eso. Un reto extra que se han planteado los estudiosos de la mente, radica en reproducir también el genio creativo: la capacidad de crear arte, música principalmente. Crear máquinas que sean capaces de reconocer la belleza en un paisaje o reírse de los chistes. Ciertamente la probabilidad de éxito de estos proyectos aún se encuentra en debate.

Finalmente, resumiendo todo lo anterior, se puede decir que un cuerpo humano, pensado como un ente físico, contiene cuatro módulos básicos:

- Un sistema para mantenerse en funcionamiento (sistema digestivo, respiratorio, circulatorio).

- Un sistema motor que le permite interactuar con su entorno, con actividades que van desde desplazarse de un lugar a otro hasta aquellas de comunicación, como el habla.

- Sensores que le permiten enterarse tanto de su estado interno (percepción del hambre, sueño, dolor, temperatura, etc.) como del mundo que le rodea (sentidos de la vista, oído, tacto, etc.) y de su propio estado con respecto a su ambiente, como sucede con el sentido del equilibrio. Por supuesto que este mismo sistema sensorial le permite percibir las acciones que realiza el sistema motor, habilitando la capacidad de retroalimentación.

Figura 2.2 Algunos de los sentidos.

- Un sistema lógico que le permite manejar toda la información que recibe a través de los sentidos y que le permite desarrollar comportamientos adecuados ante diferentes circunstancias. Éste es el encargado de coordinar las interacciones entre todos los componentes del individuo. Así mismo, tiene la capacidad de evolucionar y adaptarse a un entorno en constante cambio.

2.2 Los Agentes Inteligentes

En 1955 la Inteligencia Artificial surge oficialmente como disciplina de la Informática en la Conferencia de Computación de Dartmouth, al establecer que es posible simular inteligencia en una máquina y fijarse la meta de programar dicha simulación. Esta afirmación se expresa en el siguiente párrafo, que es una paráfrasis de la hipótesis de sistemas simbólicos físicos:

> *"Nuestro cerebro no posee un acceso directo al mundo exterior. Solamente podemos operar sobre una representación interna suya la cual se corresponde con una colección de estructuras de símbolos. Dichas estructuras pueden tomar la forma de un patrón físico cualquiera, por ejemplo un vector de interruptores eléctricos dentro de un ordenador, o un conjunto de neuronas activadas en un cerebro biológico. Un sistema inteligente (cerebro u ordenador) puede operar sobre las estructuras con el objetivo de transformarlas en otra construcción. El pensamiento consiste en la extensión, o desarrollo, de estas estructuras, descomponiéndolas y reformándolas, destruyendo algunas y creando nuevas. La inteligencia entonces no constituye nada más que la habilidad de procesar estructuras de símbolos. Existe en un entorno diferente al hardware que le da soporte, lo trasciende y puede tomar diferentes formas físicas."*[8]

En el primer renglón de esta cita es posible vislumbrar el inicio de la gran complejidad correspondiente al esfuerzo de imitar a la mente humana: El cerebro humano no posee un acceso directo al mundo exterior, sin embargo, debe construirse una representación de él, a partir de lo que perciben los sentidos y después operar y razonar en ella. El ser humano contará únicamente con esta representación para desenvolverse en su medio ambiente y las transformaciones que realice sobre ella, le indicarán las posibles consecuencias de sus actos. Este punto de partida es el que da origen al concepto de agente:

> *"Un agente es cualquier cosa capaz de percibir su medio ambiente con la ayuda de sensores y actuar en ese medio usando actuadores."* [9]

Esta definición puede ser ilustrada con el esquema siguiente:

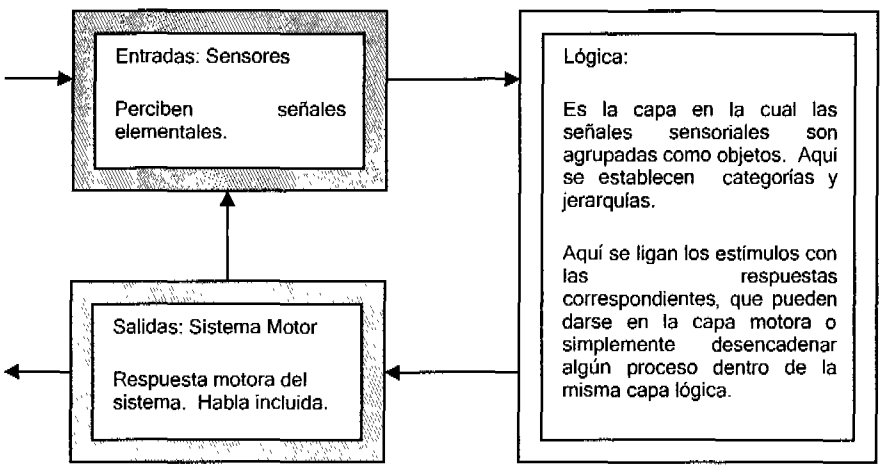


Figura 2.3 Esquema general de un agente.

Este trabajo se avoca principalmente a trabajar con la interacción entre el módulo sensorial y el módulo de la lógica. Particularmente se enfoca en el análisis de la información recibida, traduciendo las percepciones en símbolos internos que son agrupados de acuerdo a una estructura jerárquica que va siendo generada dinámicamente. Dentro del módulo lógico se tiene la meta de analizar la información recibida y almacenar los resultados en términos de reglas para un planificador. Una vez descritas de este modo, podrán ser utilizadas como definiciones de acciones para un agente completo y lo que resulta más interesante es que estas reglas habrán sido generadas a partir de experiencias observadas por medio de los sensores.

Figura 2.4 El agente de este trabajo, es capaz de aprender a partir de las acciones que observa en el mundo. (Aunque el mundo con el que se trabaja es mucho más sencillo que el mundo real). En la foto: Wakamaru (Mitsubishi Heavy Industries, Japón) [10].

Este desarrollo deberá ser realizado para un mundo extremadamente sencillo, sin embargo se espera que el estudio iniciado en esta tesis pueda servir como base y guía para trabajos futuros que funcionen para agentes más complejos y realistas.

2.3 UN ROBOT

Un robot móvil es un agente con la capacidad de percibir el medio ambiente real, a través de sus sensores, adaptarse a él e interactuar con él a gracias a las capacidades motoras que le son otorgadas. Un robot debe ser capaz de resolver problemas, mediante la planeación de las acciones necesarias para alcanzar sus objetivos.

Savage, Billinghurst y Holden diseñaron un sistema experto con diversos módulos que realizan las funciones citadas [11]. Dicho sistema fue planteado para trabajar tanto con ambientes virtuales como reales. La imagen siguiente describe su arquitectura y a continuación se incluye una descripción breve de cada uno de los módulos.

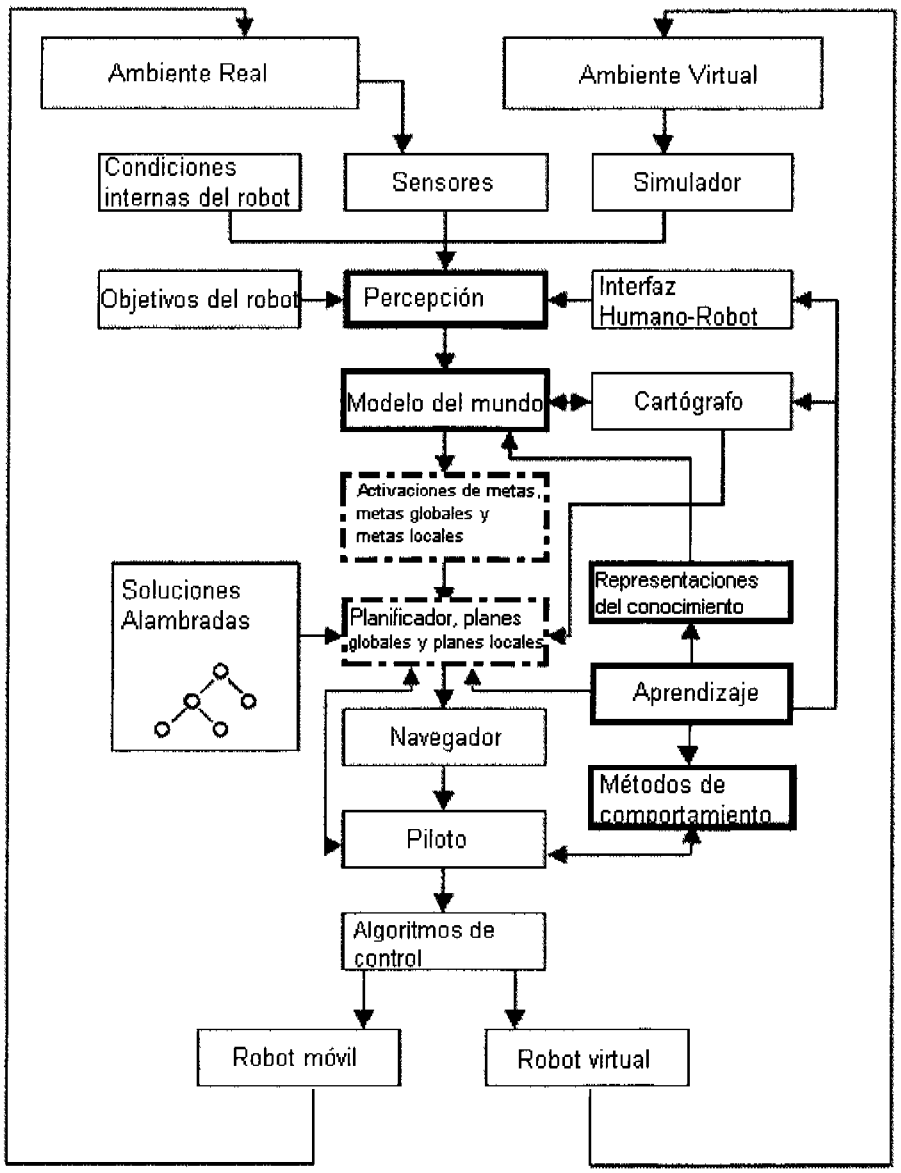


Figura 2.5 Diagrama que muestra los diversos componentes para un robot. Se encuentran enmarcados en negro aquellos que abarca el agente desarrollado en esta tesis. Los componentes que recibirán la información generada se encuentran delimitados por una línea punteada [11].

- **Ambiente real.** Mejor conocido como el mundo real, el ambiente real consiste en la parte física (paredes, objetos, luces, sonidos, etc.) con la cual interactúa un robot y que no se encuentra restringido e idealizado. Por ejemplo, el ambiente real de operación de un robot puede ser una fábrica, un edificio, una casa o un taller. El ambiente real es el lugar de operación del robot donde está expuesto a todo tipo de situaciones imprevistas.

- **Ambiente virtual.** El ambiente virtual consiste en una idealización del ambiente real de operación del robot, a menudo consiste en un ambiente simulado o ambiente controlado donde se pueden agregar uno a uno los factores externos con los que interactúa el robot (fricción, errores de

movimiento u operación, nuevos elementos u objetos, iluminación, temperatura, etc.). Las acciones de los robots son perfeccionadas primero actuando en ambientes virtuales, para luego ser probadas en el mundo real.

- **Simulador.** Cuando se trabaja con ambientes virtuales es necesario poseer un modelo que simule la realidad. Este modelo permite prever qué sucede cuando el robot ejecuta cierto tipo de acción, obteniendo información como si se tratase del mundo real. En la medida en que la simulación se asemeje a la realidad, más sencillo será implementar procedimientos dentro del robot, que funcionen bien bajo condiciones reales de operación. Por ejemplo, si se modelan las ecuaciones de propagación de ondas para los sensores o las ecuaciones de movimientos de los motores, la simulación será más realista.

- **Sensores.** Los sensores constituyen la interfaz de entrada al robot, es decir, le permiten obtener información del mundo externo (ya sea real o virtual). Entre los tipos de sensores se citan los siguientes:

  - Sensores de distancia (ultrasonido, infrarrojo, láser, etc.).
  - Sensores de visión (cámaras).
  - Sensores de contacto.
  - Sensores auditivos (micrófonos).

- **Condiciones internas del robot.** Algunos tipos de sensores están enfocados al interior del robot. Permiten hacer mediciones de información sobre la operación interna del robot. Ejemplos de ello son los medidores de nivel de batería, contadores de vueltas de las llantas, giroscopios o brújulas.

- **Interfaz hombre-robot.** Una interfaz hombre-robot permite la comunicación entre el robot y un ser humano. Por ejemplo, un módulo de procesamiento del lenguaje natural, le permitiría al robot recibir comandos hablados por parte de un operador.

- **Tareas del robot.** Las tareas que el robot debe realizar son especificadas mediante un guión o *script*. Es posible utilizar un módulo que haga uso de dependencia conceptual[12] para implementar los procedimientos a realizar. La dependencia conceptual permite representar una tarea específica, por ejemplo "robot ve a la cocina", mediante una o más primitivas de acción con la forma:

  ```
  (<nombre primitiva> (actor actor) (object objeto) (from origen) (to destino))
  ```

  Un ejemplo de ello es la primitiva *ptrans* que transfiere físicamente un objeto de posición (sinónimo de *trasladar*). Para el ejemplo citado la representación sería

  ```
  (ptrans (actor robot) (from robot) (to kitchen))
  ```

(Cabe aclarar que en este caso (from robot) indica la posición actual del robot.)

- **Sistema de percepción.** A partir de la información (en bruto) proporcionada por los sensores (por ejemplo la información relativa a imágenes de entrada o sensores de distancia), la interfaz hombre-robot (lenguaje natural) y las tareas del robot (dependencia conceptual), el módulo de percepción obtiene representaciones simbólicas de ellos y genera una creencia.

- **Modelo del mundo.** En esta etapa se validan las creencias generadas por el sistema de percepción, interactuando con el cartógrafo.

- **Cartógrafo.** Posee una representación del mundo. A menudo consiste en un mapa que puede ser geométrico (polígonos), simbólico (pared, lisa) o topológico. Cuando el modelo del mundo valida una creencia (a menudo relacionada con las condiciones externas del mundo) el cartógrafo actualiza su información y agrega o elimina los elementos representados.

- **Activación de metas globales y locales.** Una vez que se han validado las creencias se activan objetivos específicos (por ejemplo movimientos) encaminados a alcanzar las metas señaladas.

- **Soluciones alambradas.** Consisten en secciones de código que resuelven problemas específicos (por ejemplo tomar un objeto, girar cierto ángulo o avanzar cierta distancia, etc.). Por lo general consisten en máquinas de estados que realizan cierta función; sin embargo, si el robot cuenta con un módulo de aprendizaje, una solución alambrada puede consistir en una secuencia de acciones que el robot ha aprendido previamente.

- **Planificador.** Con base en las metas específicas, este módulo decide qué procedimientos con los que cuenta de antemano va a utilizar para estructurar una secuencia de acciones que permita resolver los objetivos globales o locales. Por ejemplo, contando con acciones básicas de desplazamientos del robot, encuentra el camino a seguir para llegar a una posición determinada.

  Este módulo busca secciones de código que resuelvan problemas parcialmente.

- **Navegador.** Una vez que se tiene la ruta que el robot debe seguir (puntos a visitar) el navegador encuentra las ecuaciones de movimiento: velocidad, aceleración, distancia y giro que deben aplicarse para trasladar al robot a cada una de las posiciones establecidas.

- **Representaciones del conocimiento.** Consiste en una codificación del conocimiento mediante reglas y hechos (por ejemplo dentro del sistema experto CLIPS o el lenguaje prolog). Está en contacto permanente con el modelo del mundo ya que hechos o reglas específicos pueden estar relacionados con las condiciones del mundo real.

- **Aprendizaje.** Este módulo permite que procedimientos elaborados exitosamente y situaciones previas, pasen a formar parte de la base de conocimiento. Por ejemplo un robot que no tenga mapa y pueda generar dicho mapa con observaciones propias.

- **Piloto.** Ejecuta los comandos dados por el navegador, hace que gire el robot y avance, controla los motores mediante máquinas de estados y recibe permanentemente información de los sensores.

- **Métodos de comportamiento.** Consisten en procedimientos que son activados cuando se generan ciertas situaciones, como la detección de obstáculos inesperados. En este caso del ejemplo, es necesario ejecutar un comportamiento que permita evitar el obstáculo, como el algoritmo BUGS, que sigue los contornos del obstáculo para evitarlo.

- **Algoritmos de control.** Permiten controlar dispositivos de hardware con exactitud mediante máquinas de estados, filtros adaptables, control analógico y digital retroalimentado o en lazo abierto, etc. Los algoritmos de control pueden ser aplicados tanto al robot real como al robot virtual, cuando se está simulando el ambiente real.

- **Robot virtual.** Simula el agente sobre el cual se aplican los algoritmos de control y movimiento ejecutados por el piloto.

- **Robot Móvil.** Robot real que se encuentra en contacto con el mundo externo.

Una vez que se han ejecutado acciones sobre el robot real o virtual, el resultado de sus acciones modifica el entorno real o virtual, dando origen a nuevas percepciones que a su vez generan nuevos comportamientos y así sucesivamente.

De entre todos los módulos de Virbot descritos, este trabajo se enfoca en los aspectos siguientes:

- La interacción entre las percepciones recibidas de un sensor de visión.
- El modelo del mundo que el agente se puede generar a partir de lo percibido.
- La representación interna de todo el conocimiento, percibido y adquirido, tanto de objetos como de acciones.
- El aprendizaje mediante observación y análisis de secuencias de acciones que permiten resolver problemas.

CAPÍTULO III

PLANEACIÓN Y PERCEPCIÓN

3.1 PLANIFICADORES

> *Planificación es el proceso de búsqueda y articulación de una secuencia de acciones que permitan alcanzar un objetivo.* [9]

La meta de un planificador es encontrar una secuencia de acciones que le permitan, a un agente, alcanzar una meta. Los pasos que componen un plan son acciones atómicas realizables por el agente. Estas acciones no son descritas detalladamente en términos del hardware, no describen los movimientos finos que, por ejemplo, debe realizar un robot, como "levantar el brazo 15cm, girar la muñeca 30°". En lugar de ello, **un planificador especifica acciones** en un nivel de abstracción más alto: **en términos de sus efectos sobre el mundo.**

Por ejemplo, un planificador para un robot en un mundo de bloques puede incluir acciones como "toma el objeto x" o "ve a la posición x". El micro control requerido para que el robot ejecute los planes se encuentra implícito en la definición de estas acciones de nivel alto. La figura 3.1 ilustra este concepto: SCHRDLU recibe instrucciones en un alto nivel de abstracción (Toma un bloque verde) y los movimientos detallados que debe realizar la garra quedan a cargo del motor gráfico.

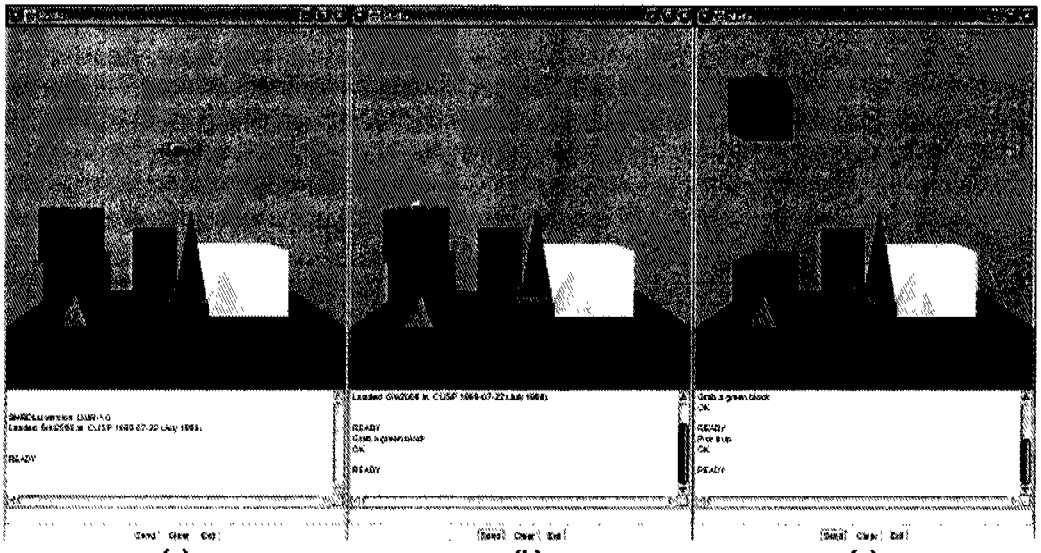


Figura 3.1 En el mundo de bloques las acciones son descritas en términos de expresiones de alto nivel. Los movimientos finos se almacenan internamente. (a) Una interfaz gráfica para SCHRDLU. (b) Comando: "Toma un bloque verde". Internamente: Indicaciones precisas para desplazar la mano a las coordenadas <100, 350, 500> y agarrar bloque B7. (c) Comando: "Levántalo". Internamente: Indicaciones precisas para desplazar la mano verticalmente 500 unidades.

De este modo, la secuencia de acciones para "Robot, tráeme el periódico, que está en el patio" podría ser:

- Ve al patio.
- Busca el periódico.
- Acércate al periódico.
- Toma el periódico.
- Vuelve conmigo.

Los planes son generados mediante una búsqueda en el espacio de acciones posibles, hasta que se descubre alguna secuencia de acciones que permite alcanzar el objetivo deseado. Este espacio contiene las representaciones de los estados del mundo que van siendo modificados conforme se ejecutan cada una de las acciones. La búsqueda termina cuando se ha producido un estado del mundo que corresponde con el estado objetivo.

Los planificadores descomponen la descripción del mundo en términos de condiciones lógicas y representan un estado como una secuencia de predicados afirmativos conectados. Sin embargo, queda abierto el problema de decidir qué tan detallada debe ser la descripción del ambiente. Para simplificar las descripciones, se utiliza la **hipótesis de un mundo cerrado**, es decir, se asume que todas las condiciones que no son mencionadas explícitamente en un estado son falsas. A partir de las afirmaciones respecto al estado actual del mundo, se determina qué acciones es posible realizar.

Además, en un planificador, una acción es especificada en términos de las precondiciones que deben cumplirse antes de ser ejecutada y de las consecuencias que se siguen cuando se ejecuta.

El problema de especificar qué es modificado por una acción y qué no lo es, es conocido como el **problema del marco**. Este tópico se vuelve más importante conforme se incrementa la complejidad espacial del problema.

La descripción de una acción no incluye especificaciones acerca de los mecanismos necesarios para que el agente pase del estado inicial (antes de la acción) al estado final (después de la acción). Esta característica permite tener un "concepto inmediato" de lo que esa acción representa, sin tener que descender a los mecanismos concretos del agente que la ejecuta.

Esta forma de descripción es, en realidad, un esquema de una acción que consta de tres elementos [9]:

- El nombre de la acción y la lista de parámetros de los que depende la acción.

- La precondición, que es la unión de predicados afirmativos sin dependencia funcional, estableciendo lo que debe ser verdad en un estado antes de que una acción pueda ser ejecutada. Todas las variables de las precondiciones deben también aparecer en la lista de parámetros de la acción.

- El efecto, que es la unión de predicados sin dependencia funcional describiendo cómo el estado cambia cuando la acción es ejecutada. Las variables en el efecto deben pertenecer a la lista de parámetros de acción. Este efecto puede ser descrito en términos de dos listas de predicados: aquellos que se volverán verdaderos después de que la acción sea ejecutada (lista "Añadir") y aquellos que se volverán falsos (lista "Borrar").

Una de estas acciones es aplicable en cualquier estado que satisfaga sus precondiciones y su aplicación consiste en agregar los predicados afirmativos en el efecto de la acción y eliminar los negativos (los hechos que dejan de ser verdad), tras haber sustituido las variables en dichas precondiciones por los valores del estado que las satisface. Cuando un efecto positivo de una acción ya se encuentra presente en la descripción del estado, éste no es agregado nuevamente; así mismo si un efecto negativo no estaba, esa parte del efecto es ignorada. Como consecuencia, cada predicado no mencionado en el efecto permanece sin modificar.

La solución a un problema de planificación, en su forma más sencilla, consiste en una secuencia de acciones que, ejecutada a partir del estado inicial, da como resultado un estado final que satisface un objetivo. Sin embargo, algunos problemas admiten varias soluciones que están conformadas por conjuntos de acciones parcialmente ordenadas, tales que cada secuencia de acciones que respete el orden parcial es una solución al problema. Cuando el sistema detecta primero los órdenes parciales, y después obtiene un plan general a partir de estas subsecuencias, se dice que realiza **planificación no lineal**.

3.1.1 Máquinas de Inferencias

Una máquina de inferencias es un sistema que incluye:

- Un lenguaje en el cual es posible expresar las descripciones de los estados y las definiciones de diversas acciones en forma de reglas y

- Un motor de inferencias capaz de reconstruir al menos una secuencia de acciones que resuelva el problema de planificación, cuando detecte que las precondiciones han sido activadas, obedeciendo a las reglas que fueron registradas con anterioridad.

CLIPS es la máquina de inferencias para la cual se generarán las reglas del experimento a desarrollar en esta tesis. En el apéndice A se describe este sistema concreto.

3.2 EL ESPACIO DE BÚSQUEDA

Se ha dicho que un planificador debe realizar una búsqueda sobre el espacio de acciones posibles. En esta sección se profundiza en el desarrollo de esta etapa.

El mundo de cubos es un ejemplo de entorno muy útil para ilustrar el funcionamiento de un planificador. El mundo de cubos consiste en un ambiente con una mesa, sobre la cual se colocan cubos que pueden ser apilados. El agente puede agarrar los cubos que se encuentran despejados, levantarlos y llevarlos a otras posiciones. Concretamente, los cubos solamente pueden ser colocados sobre la mesa o sobre otro cubo (sobre la superficie de uno exactamente). No se admiten posiciones más complejas, como un cubo sobre dos cubos o torres de cubos rotados, como se muestra en la figura 3.2.

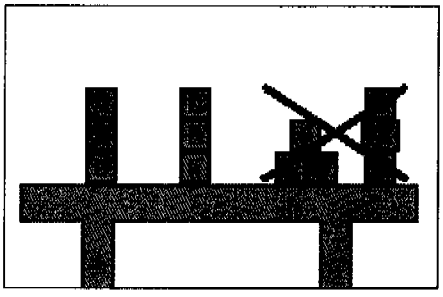


Figura 3.2 Una configuración del mundo de bloques. Sólo se puede apilar un cubo sobre la mesa o sobre la superficie de otro cubo.

Dadas estas condiciones, es posible describir el estado del mundo en términos de los siguientes predicados:

$Posición(x, y, z)$	El bloque w se encuentra en las coordenadas x, y, z.
$Sobre(x, y)$	El bloque x se encuentra inmediatamente sobre el bloque y.
$Libre(x)$	El bloque x no tiene nada sobre él.
$Sosteniendo(x)$	El brazo del robot se encuentra sosteniendo al bloque x.
$Sosteniendo()$	El brazo del robot no sostiene nada.
$Sobre\_la\_mesa(w)$	El bloque w se encuentra sobre la mesa.

El predicado $Sobre\_la\_mesa(w)$ es una forma abreviada para el predicado $Posición(x, y, z)$, donde z es el nivel de la mesa. Análogamente, $Sobre(x, y)$ indica que el bloque x se encuentra en una posición tal, que las coordenadas de su base coinciden con la superficie superior del bloque y.

Es necesario especificar, que existe un cierto número de relaciones implícitas en la descripción dada del mundo de bloques. Entre ellas se incluyen las siguientes:

1. $\forall_x \neg \exists_y Sobre(y,x) \Leftrightarrow Libre(x)$
2. $\forall_y \forall_x Sobre\_la\_mesa(y) \Rightarrow \neg Sobre(y,x)$
3. $\forall_y Sosteniendo() \Leftrightarrow \neg Sosteniendo(y)$

El primer enunciado establece que: el bloque x se encuentra libre, cuando no existe un bloque y tal que y esté sobre x. En términos de procedimientos, este enunciado implica que: "para librar al bloque x, se debe remover cualquier bloque y que se encuentre sobre él". La segunda relación dice que, si cualquier bloque se encuentra sobre la mesa, no se encuentra sobre ningún otro bloque.

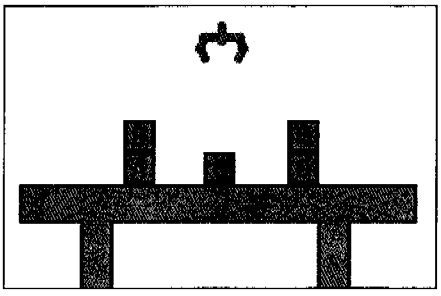


Figura 3.3 Una configuración del mundo de bloques.

De este modo, el mundo de bloques de la figura 3.3 queda descrito con la conjunción de los siguientes predicados:

$Sobre\_la\_mesa(a)$	$Sobre(b,a)$	$Libre(b)$
$Sobre\_la\_mesa(c)$	$Sobre(e,d)$	$Libre(c)$
$Sobre\_la\_mesa(d)$	$Sosteniendo()$	$Libre(e)$

En este mundo virtual, el robot puede ejecutar las acciones siguientes:

$Ir\_a(x,y,z)$	Mover la mano mecánica a la posición indicada por las coordenadas x, y, z. Esta posición puede encontrarse implícita en la ejecución de los comandos siguientes. Por ejemplo, $Levantar(w)$ requiere que el brazo se dirija a las coordenadas de w.
$Levantar(w)$	Levantar al bloque de su posición actual y sostenerlo. Para ejecutar esta acción se requiere que el bloque no tenga nada encima, que la mano mecánica se encuentre libre y que la computadora conozca la posición de w.

$Bajar(w)$	Colocar al bloque w en alguna posición sobre la mesa y guardar su nueva localización. La mano mecánica debe estar sosteniendo a w.
$Apilar(u,v)$	Colocar al bloque u sobre el bloque v. La mano mecánica debe estar sosteniendo a u y v no debe tener bloques encima.
$Desapilar(u,v)$	Quitar al bloque u de encima del bloque v. U no debe tener bloques encima, v debe tener a u sobre él y la mano debe estar libre antes de ejecutar este comando.

Dadas las operaciones que se quieren realizar con los cubos, es posible omitir la regla $Ir\_a(x,y,z)$, ya que ésta será invocada por las demás.

Dadas las representaciones de los estados y definidas las operaciones que se pueden ser realizar, se plantea el problema siguiente:

> *Dado un estado inicial, encontrar alguna secuencia de acciones que produzca un estado final deseado.*

La búsqueda queda planteada en términos de las acciones que se pueden efectuar sobre el estado actual y los estados que se generan a partir de su aplicación.

La figura 3.4 muestra una parte del árbol de búsqueda generado a partir del estado ilustrado en la figura 3.3. Los nodos hijos, contienen los estados que se alcanzan tras ejecutar cada una de las reglas aplicables, sobre el estado en el nodo padre. La figura también ilustra cómo es posible tener caminos que devuelvan el mundo a algún estado anterior. Generalmente, la búsqueda de la solución requiere eliminar estos caminos.

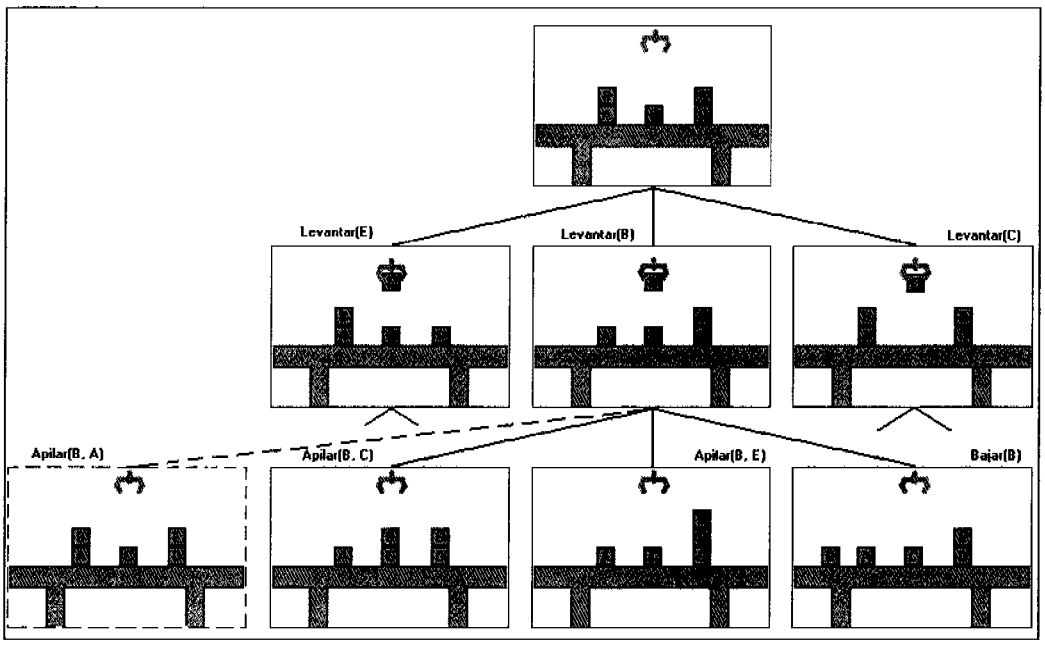


Figura 3.4 Vista parcial de los estados generables a partir de la realización de las acciones posibles en cada uno de ellos. Sobre esta gráfica se buscan los planes para alcanzar los estados deseados.

En este mundo, un plan para alcanzar un estado objetivo, queda definido por alguna secuencia de varias acciones *Levantar, Bajar, Apilar* y *Desapilar* que vaya transformando el estado del mundo hasta alcanzar el estado objetivo.

En general, el problema de la planificación involucra realizar una búsqueda sobre el espacio de estados generables por cada acción posible. El algoritmo utilizado para recorrer el árbol determinará qué solución se encuentra, si es que se encuentra alguna.

Por ejemplo, se puede realizar una búsqueda por amplitud, la cual podría permitir encontrar la solución que requiere menos pasos; o una búsqueda en profundidad que, si encuentra una solución en las primeras ramas que explora, es más rápida y además utiliza menos memoria. Por otro lado, es posible que el espacio de búsqueda sea tal, que cualquiera de estos algoritmos caiga en ciclos infinitos y nunca encuentre una solución e incluso sea incapaz de decidir si es que ésta no existe.

Existen otros algoritmos que pretenden subsanar algunas debilidades de estos dos primeros algoritmos, como las búsquedas en profundidad y amplitud iterativa; o las búsquedas en grafos, que detectan cuándo se ha regresado a un estado visitado previamente.

CLIPS es una herramienta para desarrollar y producir sistemas expertos. Provee un ambiente completo para la construcción de sistemas expertos basados en reglas y/o objetos [13]. En ella es posible expresar los diferentes estados en que se encuentra el mundo y las reglas de comportamiento que aquí se han explicado. A partir de esta información, CLIPS es capaz de realizar automáticamente la búsqueda, a través del espacio de estados, del

correspondiente plan a ejecutar. Inclusive, es posible seleccionar diferentes algoritmos de búsqueda, lo cual deriva en la obtención de planes distintos.

De esta manera, si un agente es capaz de extraer las reglas y escribirlas en la sintaxis de CLIPS, ésta herramienta será capaz de generar planes a partir de ellas, para resolver problemas futuros.

3.3 El Problema Base

A partir de este capítulo se estudiará el proceso de recepción de información, generación de la representación interna y extracción de reglas de comportamiento para un problema concreto, pero representativo.

Existe una gran cantidad de actividades que resultan especialmente sencillas para los seres humanos; tanto, que a menudo es difícil percatarse de todos los procesos que involucra el realizarlas. Un ejemplo es la actividad de vestirse y una parte de ello consiste en ponerse los calcetines y los zapatos.

Pareciera que ponerse el calzado es un problema trivial, sin embargo, no es igualmente trivial hacer que una computadora extraiga las reglas importantes para realizar esta actividad en el orden correcto.

Una primera solución a este problema consiste en agregar específicamente información acerca de qué es un pie, calcetín y zapato, y mostrarle a la computadora una secuencia en la cual una persona vaya poniéndose estas prendas en el orden correcto y registrarla. De este modo, si se le pidiera posteriormente a la computadora una solución al problema, podría devolver la que se le mostró. Sin embargo, cuando se le solicitara generar una solución alternativa o reconocer la acción realizada, si ésta fuera realizada en un orden ligeramente distinto (como poner primero calcetines y luego zapatos o vestir primero un pie y luego el otro) la computadora sería incapaz de seguir la acción.

En estos casos, es posible ingresar manualmente reglas a un sistema experto que expliciten el hecho de que lo único importante es que el calcetín sea puesto antes que el zapato en el pie respectivo. Cada una de estas reglas, corresponde a la ejecución de alguna acción.

Dadas las reglas, obtener alguna solución se convierte, una vez más, en un problema de planificación, que requiere de realizar una búsqueda. Dicho problema, en general, es conocido como **planificación ordenada parcialmente (POP)**, pues para definir un plan válido, sólo es necesario sujetarse a los requerimientos de un ordenamiento parcial entre las reglas aplicables. Dicho de otra manera, en este tipo de problemas de planeación, no existe un orden único en el cual se deban ejecutar las acciones para llegar al estado meta: existen varios.

Existen algoritmos POP que sacan ventaja de este hecho para obtener los planes en forma más eficiente. Éstos son explicados por Russell y Norvig en

[9], en el capítulo correspondiente al tema, y utilizan como ejemplo, precisamente, el problema de ponerse los zapatos.

Sin embargo, lo que esta tesis explora, es la posibilidad de que un agente inteligente genere automáticamente estas reglas a partir de su experiencia. Para ello, es necesario analizar porqué un ser humano puede hacerlo y cómo lo hace.

3.3.1 El Problema Base en el Mundo Real

Aunque para una persona adulta resulte inmediato saber cómo vestirse, es conveniente recordar que no siempre fue así.

Para que un niño pequeño aprenda a colocarse su indumentaria en la forma correcta, es necesario que alguna persona mayor le muestre paso a paso lo que tiene que hacer. En este proceso es fundamental el tipo de información que recibe el niño a través de sus sentidos.

Dependiendo de la edad del niño, lo más seguro es que la persona mayor tome las manos del niño y con ellas tome un calcetín, para luego colocarlo en el pie del niño. A la edad a la que el niño ya puede aprender a calzarse, ya debe encontrarse familiarizado con la secuencia de acciones que se suceden con este fin; con la presencia de otro calcetín y de los zapatos. Durante la acción de ponerse el primer calcetín, solamente enfocará sus sentidos sobre los movimientos inducidos de sus manos y pie, la textura del calcetín cubriendo su pie y la sensación final de soltar todos los elementos, dejando el pie cubierto (lo cual ve y siente). Más allá de los movimientos finos que el niño debe aprender (y que le tomarán más tiempo), su análisis del proceso como un todo puede darse más rápido.

A continuación, puede ocurrir alguna de dos cosas: que le sea colocado el zapato para el mismo pie, en cuyo caso es incluso probable que el pequeño adivine que el proceso se repetirá idénticamente para el otro pie. De algún modo no debe tomarle mucho tiempo darse cuenta de las semejanzas entre los miembros del lado derecho con los del izquierdo.

Si, por el contrario, se continúa colocándole el otro calcetín, antes que el zapato, es muy probable que no adivine lo que hay que hacer a continuación con los zapatos, sino hasta el último paso, debido a que colocar algo más y diferente a lo anterior sobre los pies ya cubiertos, podría no ser tan obvio.

En cualquiera de los dos casos, lo más seguro es que, una vez aprendidos los objetivos que debe alcanzar (calzar calcetines y zapatos), nadie deba mostrarle otro orden para hacerlo y, sin embargo, podría empezar a usar otro por sí mismo y no tendrá ningún problema en identificar la misma acción en cualquier otra persona que utilice otro orden. Esto sucederá porque inconscientemente habrá seccionado el problema en "poner calcetín" y "poner zapato sobre el pie con calcetín"; esto sucede mediante la detección de acciones que se repiten múltiples veces (como la secuencia: agarrar el objeto con ambas manos, colocarlo sobre la extremidad inferior y soltar), así como

de los objetivos parciales "cubrir con el calcetín" y, sobre lo anterior, "cubrir con el zapato".

Algunos niños, tal vez, también intentarán experimentar con colocar el zapato sin calcetines. Esta puede ser una manifestación de un olvido o de simple curiosidad, pero será necesario abstenerse de continuar el análisis por esta rama ya que el trabajo podría extenderse más de lo deseado en este momento.

Lo que se plantea a continuación es: cómo hacer que una máquina reproduzca el proceso de aprendizaje expuesto.

3.3.2 Otros problemas

Por completez, es conveniente mencionar algunos problemas que podrían ser tratados con estos razonamientos.

Puede citarse, por ejemplo: el problema de preparar un pastel. Para prepararlo se requiere satisfacer varios prerrequisitos: conseguir los ingredientes (piénsese, por el momento, que no es necesario comprar nada) y luego seguir los pasos de la receta. A menudo, varios de estos pasos pueden ser ejecutados, algunos, en órdenes distintos, otros, en algún orden riguroso. Por ejemplo: puede sacarse primero la mantequilla del refrigerador o el harina del alacena, etc. En la práctica algunas decisiones podrían verse influenciadas por factores de conveniencia como tomar primero todo lo que está cerca, sin embargo, estos puntos no se cubren por ahora. El resto del problema puede ser planteado en el sistema propuesto.

Otro ejemplo, puede ser arreglar la mochila para un día de escuela. ¡Hay muchas posibilidades para meter los útiles! Aunque este requeriría de un sistema de representación espacial mucho más refinado.

Finalmente, para no hacer muy extensa la lista, se puede mencionar el problema completo de vestir a una persona.

En general, se está hablando de extraer reglas para problemas que permiten planificación no lineal. En particular, para los alcances de este trabajo, se trata de problemas de alcance local, es decir, aún no se incluyen conceptos más abstractos como "tienda", "escuela" que requieren que se complete el uso del sistema de clases propuesto; punto que aún no se aborda en esta tesis.

3.4 Acciones a Considerar

El primer problema con el que debe enfrentarse el agente computacional es el paso de la información acerca del mundo exterior, percibida a través de sus sensores, a una representación interna sobre la cual pueda trabajar.

El problema de seleccionar una estructura interna, en la cual representar la información recibida, es un problema abierto para el caso general, pero puede

ser resuelto para problemas concretos. Para comenzar con este trabajo, resulta imperativo restringirse a una representación muy sencilla tanto de la información recibida como de su interpretación correspondiente.

Para una primera propuesta, se considerarán tres acciones fundamentales que pueden observase en las imágenes de interacciones entre objetos:

- Asir objetos (unir)
- Soltar objetos (separar)
- Poner objetos (meter)

(Se omite el tratamiento de desplazamientos, aunque en principio sea posible incluirla posteriormente.)

Dado que se desea adquirir la información acerca del mundo exterior a través de sensores, es necesario considerar qué tipo de información pueden proveer estos sensores, de modo que sea posible identificar estas acciones.

En la descripción del problema se mencionó el uso de tres sentidos: vista, tacto y aquel que permite al individuo saber en qué posición se encuentran sus miembros. Sin embargo, para simplificar el problema, es posible utilizar a un agente que sólo posea un sistema de visión. Este sistema debe ser capaz de seguir las acciones realizadas por otro individuo.

3.5 La Simulación de un Sistema de Visión

Un robot inteligente deberá tener un sistema de visión que incluye la captura de la imagen (digitalización) y segmentación. A partir de la imagen segmentada, se podrán reconocer diversos objetos.

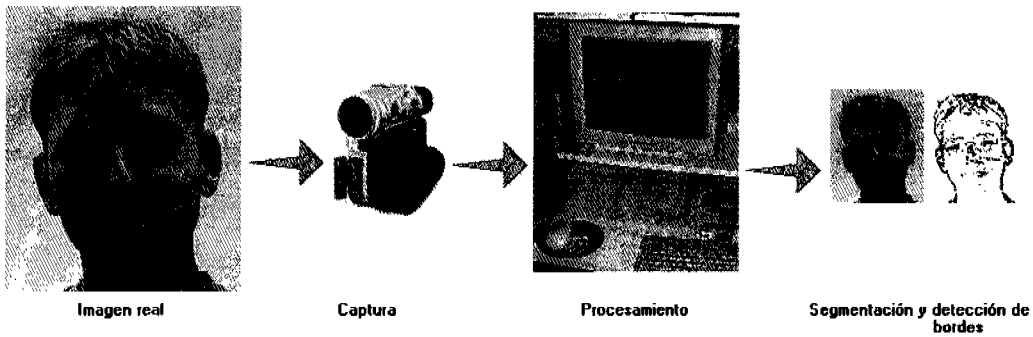


Figura 3.5 Captura y procesamiento de las imágenes provenientes del mundo real [14].

Una vez capturada la imagen, se puede proceder a analizarla. Existen varias técnicas para procesarlas, así como criterios de segmentación y algoritmos de detección de bordes. Se denomina **clasificación** al proceso de extracción de información temática a partir de imágenes. El objetivo final de este proceso es asignar una membresía de clase a todos los píxeles de una imagen. Si es posible que cada clase corresponda a regiones características

de objetos (en forma semejante a como serían clasificados por un ser humano), se habrán cubierto los precedentes requeridos para implementar los algoritmos propuestos en este trabajo.

Como se expone en [21], algunas técnicas de segmentación pueden ser clasificadas en:

- Segmentación basada en fronteras.- Se utilizan algoritmos, conocidos como detectores de bordes, para extraer conjuntos de bordes a partir de una imagen. Los bordes son cambios locales significativos en la intensidad de la imagen, que usualmente están asociados con una discontinuidad en la función de intensidad o en su derivada y pueden ser producidos por diversos factores físico-geométricos como sombras, texturas, cambios en la reflectancia de los objetos, etc. Posteriormente, los bordes detectados son combinados en cadenas de bordes, que puedan corresponder con fronteras de objetos en la imagen.

- Segmentación basada en regiones.- Los píxeles son asignados a regiones según algún criterio que los distingue del resto de la imagen. Dos principios muy importantes para realizar la segmentación son el valor de similitud (intensidad, niveles de gris, compactificación de una región, etc.) y la proximidad espacial. Frecuentemente ocurre que las regiones detectadas no corresponden exactamente con objetos, en estos casos se utiliza conocimiento dependiente del dominio, para refinar la correspondencia. (Se espera que el contexto introducido por los análisis de esta tesis, pudieran, en un futuro, auxiliar en este refinamiento).

  - Segmentación basada en umbralización.- Ésta técnica se basa en el uso de histogramas de frecuencia. En un histograma de frecuencias se cuenta cuántos píxeles hay en la imagen para cada valor de la intensidad. Ésta técnica es útil cuando las frecuencias tienden a acumularse alrededor de valores definidos, de manera que los picos en el histograma corresponden a regiones en la imagen. Se elige un valor umbral como frontera entre las regiones y es utilizado para determinar a qué región pertenece cada píxel.

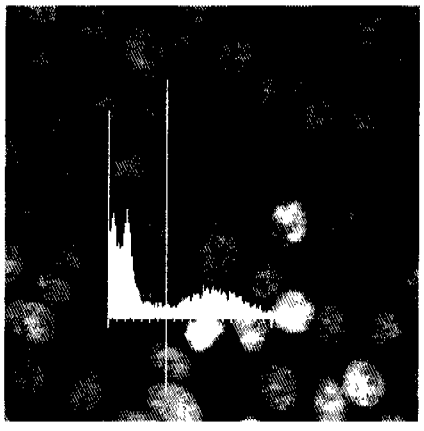


Figura 3.6 Preparación con núcleos que presentan un valor de gris claro sobre un fondo oscuro. Se visualiza el histograma de distribución de niveles de gris dónde se puede observar que éste es bimodal, con dos picos bien diferenciados, el primero corresponde al fondo y el segundo a los núcleos.

- Segmentación orientada a regiones.- Este método construye las regiones directamente; sin intentar, primero, encontrar los bordes. Se establece un criterio de homogeneidad para determinar si secciones adyacentes deben o no pertenecer a la misma región. Las regiones obtenidas finalmente deben ser homogéneas y maximales, donde maximal significa que los criterios de homogeneidad no se cumplirán tras la unión de una región con alguna de sus adyacentes.

Existen, además, técnicas que recurren al uso de otro tipo de información, como las diferencias entre dos imágenes consecutivas de objetos en movimiento. Un ejemplo es el trabajo de tesis de Darío Peregrina [22]

Todas éstas tienen como finalidad última detectar objetos en imágenes, sin embargo algunas resultan más apropiadas para algunos casos, que otras. A veces, es necesario combinar aspectos de varias de ellas e información adicional del medio ambiente, para obtener resultados satisfactorios. A continuación, se presenta un ejemplo que podría proporcionar la información requerida para el sistema aquí desarrollado.

Para efectos de ilustrar en forma más tangible la conexión entre la segmentación de imágenes y el punto de partida de este trabajo, se puede utilizar como ejemplo el método JSEG propuesto por Yining Deng y B.S. Manjunath [15]. JSEG separa el proceso de segmentación en dos fases: cuantización de colores y segmentación espacial.

En la primera etapa, los colores en la imagen son cuantizados definiendo clases representativas que pueden ser utilizadas para caracterizar regiones en la imagen, sin considerar, en absoluto, la información espacial. A continuación, los píxeles de colores en la imagen son reemplazados por su clase, obteniéndose un mapa de clases de la imagen.

En la segunda etapa se utiliza la información espacial. A partir de la aplicación de un cierto criterio local, se pueden utilizar las diferencias de intensidad entre las clases de colores, para detectar bordes. Haciendo uso de técnicas de crecimiento de las regiones se obtiene una buena segmentación en cada imagen y se logra el seguimiento de las regiones en movimiento (video).

La figura 3.7 muestra el resultado de uno de sus experimentos.

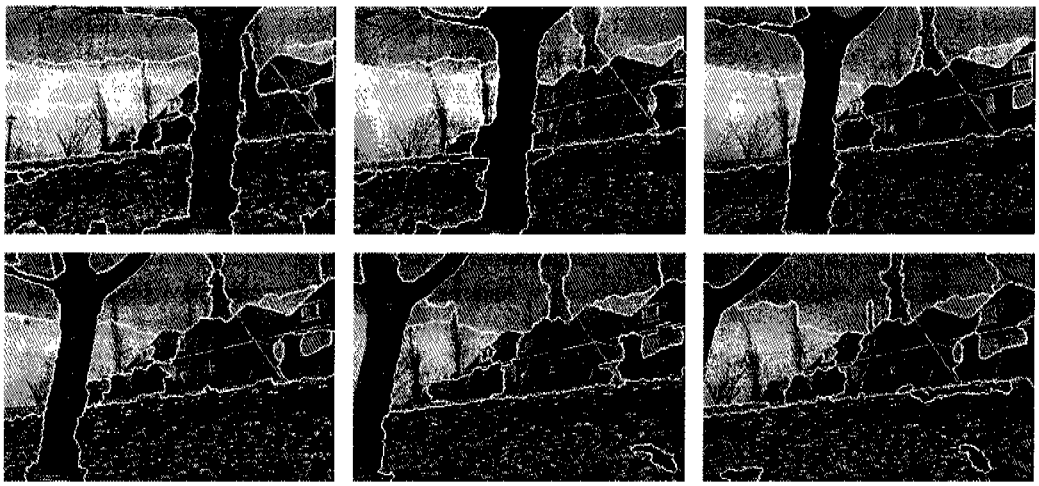


Figura 3.7 Segmentación y seguimiento en imágenes de video utilizando JSEG.

El utilizar colores y posición para generar las primeras clases, permitiría utilizar estas mismas clases para clasificar a los objetos identificados en términos de conceptos que le son naturales a los seres humanos. Como se verá más adelante, esta información podría ser manipulada con los principios que se desarrollan a lo largo de este trabajo y son fácilmente adaptables a la forma de representación del conocimiento del sistema desarrollado. Los algoritmos podrían trabajar sobre ella.

Dados estos resultados, es razonable asumir que se cuenta con un sistema de visión capaz de realizar esta primera parte del análisis.

No solamente se ha conseguido segmentar imágenes, sino que también se ha avanzado en la selección de regiones distintivas que permiten reconocer determinados objetos, incluso en posiciones distintas. Aunque estos algoritmos son aún imperfectos, nuevamente es posible, para los fines de este trabajo, comenzar asumiendo que se cuenta con un sistema capaz de identificar algunos objetos de uso común.

Andras Ferencz, Erik Learned-Miller y Jitendra Malik [16] han desarrollado métodos para seleccionar características locales especiales que permiten identificar objetos en imágenes distintas. Su sistema, aprende a seleccionar características sobresalientes a partir de pares de fotografías previamente clasificadas como "el mismo" o "diferente". Posteriormente, el sistema es capaz de tomar una sola fotografía de algún automóvil o persona, seleccionar sus características especiales y compararlos con otras fotografías, para determinar si se trata del mismo objeto o no.

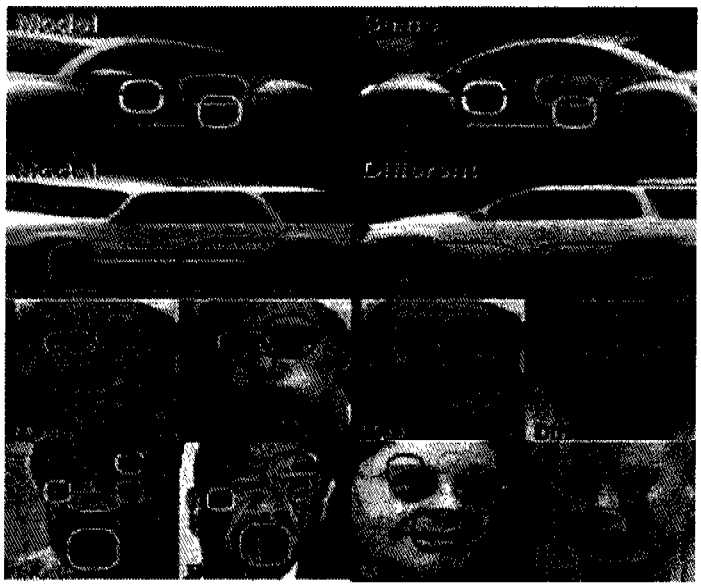


Figura 3.8 Clasificación de modelos basados en un ejemplo. Cada par de imágenes muestra un modelo y una imagen de prueba, que ha sido etiquetada como "same" (el mismo) o "different" (diferente) por el algoritmo. Las regiones que fueron utilizadas para realizar la calificación de las imágenes de prueba aparecen enmarcadas, los tonos dependen de la importancia (y orden) en que fueron utilizadas. La imagen muestra tanto pruebas exitosas como fallidas [17].

Amoonzegar Farid[23] hace una revisión del uso de sistemas basados en redes neuronales para realizar el seguimiento.

Inclusive, en la actualidad, se están desarrollando diversos métodos que, además de identificar objetos, como las partes del cuerpo (cara, manos, etc.) pueden darle seguimiento a sus movimientos, como.

Algunos de estos sistemas ya son capaces de identificar algunas acciones sencillas, como el de Heap, Tony & Samaria y Ferdinando[], que reconoce gestos y movimientos de las manos, y la tesis de maestría de Wolf Schaarshmidt [18]. Schaarshmidt logra identificar las acciones: saludar, aplaudir, levantarse, sentarse y caminar; utilizando filtrados con un modelo de piel, un algoritmo de K-medidas, cuantización vectorial y modelos ocultos de Markov.

Figura 3.9 Seguimiento del movimiento de la mano. (Imagen tomada de Schaarshmidt 2004).

No es el objetivo de esta tesis desarrollar sistemas como estos. Se hace mención a algunos ejemplos para sustentar la viabilidad de que, en un futuro, se satisfaga la hipótesis de la cual parte este trabajo: que se cuenta con un sistema de visión capaz de realizar los siguientes pasos:

- Capturar la información visual en forma de video digital.
- Segmentar las imágenes.
- Seleccionar e identificar segmentos relevantes.
- Identificar algunos objetos y determinar relaciones de adyacencia entre ellos (esta información se puede adquirir con relativa facilidad dada la posición estimada de los objetos en el espacio y algunos criterios de agrupamiento). No se requiere que la máquina sepa "qué son" esos objetos.
- Seguir el movimiento de esos objetos en el espacio.
- Entregar al módulo siguiente una representación simbólica de los elementos identificados en la imagen, sus posiciones y desplazamientos.

En principio, los objetos identificados no necesariamente deberían corresponder con objetos de nuestro lenguaje simbólico (por ejemplo: una taza, una pluma, un perro). Los "objetos" pueden ser simplemente segmentos identificables en forma consistente, como "suela del zapato", "mechón blanco en el cabello", "todo el uniforme negro vestido por un árbitro", etc. Es a partir del análisis de las modificaciones observadas en la apariencia

y/o posición de estos segmentos, que ellos o un grupo de ellos, adquirirán un significado.

Sin embargo, para hacer más clara la exposición del algoritmo que se desarrolla, en los experimentos se utilizarán segmentos primarios que efectivamente correspondan con objetos identificados por las personas, ya con una connotación semántica. Por ejemplo: zapatos, calcetines, manos, etc.

Un sistema capaz de identificar y seguir objetos, se conectaría como un módulo para este agente, quedando encargado de traducir la información visual en simbólica. Cada objeto (imagen) identificado, recibiría un símbolo que lo identifique y su posición en el mundo. Para que los algoritmos desarrollados aquí, funcionen, se requiere que el sistema de visión discretice las posiciones, privilegiando el reconocimiento de qué objetos se encuentran en posiciones adyacentes.

Existe, sin embargo, un requisito más que se le impondrá al sistema de visión y que, hasta ahora, no ha sido estudiado directamente. Es necesario que el sistema, además de identificar imágenes, sea capaz de catalogarlas de acuerdo a sus similitudes. Por ejemplo, se desea que incluya tanto a la mano izquierda como a la derecha, dentro de una misma categoría.

La figura 3.10, ilustra cómo se esperaría que clasificara, aproximadamente, algunas de las regiones indicadas en la foto.

a. 

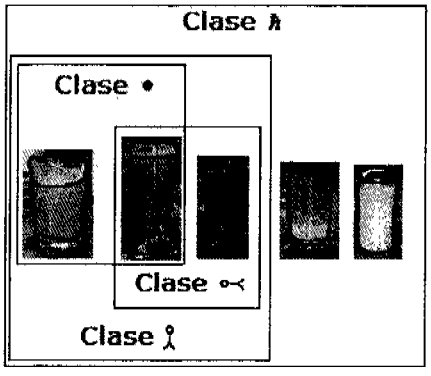
b. 

Figura 3.10 a) El módulo de visión debe seccionar la imagen recibida en regiones delimitables, reconocibles y cuyas transformaciones pueda seguir. b) Clasificación aproximada esperada para los objetos similares. Del mismo modo, se esperaría que todas las caras humanas quedaran dentro de la misma clase; similarmente para las manos.

De este modo, para analizar cualquier problema, bastará con dar una descripción de cada imagen percibida por el agente, tal que permita realizar el seguimiento de las acciones que se suceden.

Dados los símbolos que identifican a los objetos y sus posiciones en el espacio, cuadro por cuadro, este agente será capaz de realizar el análisis de las acciones complejas que se le presenten.

CAPÍTULO IV

REPRESENTACIÓN DEL CONOCIMIENTO

4.1 NOTACIÓN EN SÍMBOLOS

A continuación se define la notación en símbolos que entiende el sistema desarrollado. Esta notación ha sido diseñada para satisfacer las necesidades esenciales de expresividad para el problema a tratar y está inspirada en la notación para listas del lenguaje de programación LISP. Con la finalidad de realizar este estudio sobre casos extremadamente sencillos (y por lo tanto, manejables) solamente se definen los elementos indispensables.

De la información extraída por el sistema de visión, se utilizará la siguiente:

- Identificación de siluetas (segmentos).
- Jerarquía de imágenes, catalogadas de acuerdo a sus similitudes.
- Información posicional, sólo se tomará en cuenta si los objetos son adyacentes (se encuentran en contacto) o no.
- Seguimiento a través del tiempo (cuadro por cuadro).

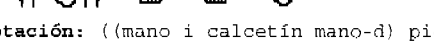
La forma de expresar esta información, será por medio de listas de objetos anidadas. Aquellos objetos que pertenezcan a una misma lista corresponderán a objetos que se encuentran en contacto físico. Inicialmente, no se consideran objetos compuestos, como sería el caso de un ser humano que tiene miembros (manos, pies, cabeza, etc.). Se comienza por considerar manos y pies como objetos atómicos independientes. Por simplicidad, no se incluyen descripciones de los objetos complejos completos, ni siquiera como conjunción de objetos atómicos adyacentes, sólo se citan a los miembros que toman parte en la secuencia que se desea estudiar.

En la figura 4.1 se ilustra una descripción del estilo de las que debe entregar el módulo de visión. En la parte superior algunos íconos (dibujos de manos, pies, etc.) permiten ver qué objetos se encuentran presentes. En la parte etiquetada "notación" se describe la línea de texto tal y como se introduce al sistema. Cada palabra corresponde a una silueta identificada en la imagen.

El sistema detecta los siguientes objetos independientes:

Notación: (mano-i mano-d pie-d pie-i calcetín calcetín zapato-d zapato-i)

El sistema detecta tres elementos atómicos en contacto, por lo que introduce un grupo, denotándolo con una lista anidada:



Notación: ((mano i calcetín mano-d) pie-d pie-i calcetín zapato-d zapato-i)

Figura 4.1 Notación para la descripción de las imágenes que serán alimentadas al agente.

Obsérvese que la descripción inicial de la imagen no establece distinción alguna entre un calcetín y otro, en correspondencia a lo que ocurriría, en principio, con un par de calcetines nuevos: no es posible distinguirlos.

En principio, los objetos ocupan una posición concreta en el espacio, sin embargo, por simplicidad, se evitará tomar en cuenta las tres coordenadas espaciales específicamente, dado que esto requeriría enfrentar el problema de la visión y los ambientes virtuales con mucha mayor precisión y ese tema, por sí mismo, puede dar origen a otra tesis. En lugar de ello, la descripción espacial se limita a los conceptos "cerca" (o en contacto, pertenecientes a un grupo) y "lejos" (aquellos objetos que no se encuentran en contacto). Estos conceptos manejan, más bien, la ubicación relativa de los objetos, la cual resulta mucho más expresiva para el problema a tratar.

De este modo, los conceptos asir y soltar se corresponden con representaciones sucesivas en las que se forman o separan grupos de objetos.

Para ilustrar la acción de "poner" o "meter" se requiere expresar mejor lo que se observa en el video. El concepto de meter, desde el punto de vista de la visión, se encuentra asociado a la desaparición de partes de una imagen. Al meter un objeto en un jarrón "se ve" cómo el objeto va desapareciendo paulatinamente "tras" el lado visible del jarrón. Cuando alguien se pone un calcetín, se ve al pie color carne ser cubierto por la textura del calcetín, mientras que el objeto calcetín va tomando una consistencia más rígida, adherida a la forma del pie en el cual se está colocando.

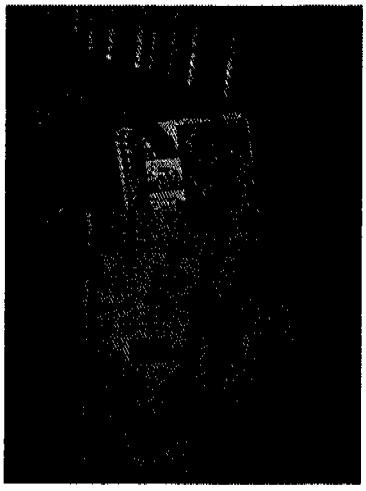


Figura 4.2 Cuando se mete un objeto dentro de otro, se observa como una imagen va quedando oculta tras otra.

Esta acción sería observada por el sistema de visión a lo largo de varios cuadros. En principio, el sistema de visión debería poder generar esta descripción cuadro por cuadro. Sin embargo, realizar esta descripción en texto, cuadro por cuadro, resultaría extremadamente engorroso. Además, dado que, por el momento las descripciones están siendo generadas manualmente, es más conveniente introducir un sistema que resuma todos estos cuadros en dos líneas. Estas dos líneas, igualmente pudieran ser

sintetizadas directamente por el sistema de visión, de modo que la descripción de la acción "meter" sea entregada a este módulo en los siguientes términos:

- En la primera línea, se indica que los dos objetos participantes entran en contacto. Recordar que la unión de dos elementos se indica haciéndolos miembros de una misma lista, por ejemplo: *(cuaderno maleta)*.

- En la segunda, se indica cómo lucen los dos objetos una vez que uno ha sido introducido en el otro. Para ello se introduce el símbolo ->. A la izquierda se encuentra la pareja original, a la derecha, lo que se ve ahora en lugar de ella. Por ejemplo: *(cuaderno maleta) -> maleta*, en este caso, tras meter el cuaderno, ya sólo es visible la maleta.

La figura 4.3 ilustra esta descripción para el caso en que ambas manos son utilizadas para poner un calcetín. En la primera línea el pie y el calcetín deben ser adyacentes (pertenecer al mismo grupo). En la segunda se indica cómo el grupo (pie-i calcetín) se convierte en cpi (un pie con calcetín).

Las manos juntan al pie izquierdo y un calcetín:

((mano-i (pie-i calcetín) mano-d) pie-d calcetín zapato-d zapato-i)

Las imágenes del pie y el calcetín se unen, dando origen a una nueva imagen, que tiene características de ambos:

((mano-i ((pie-i calcetín) -> cpi) mano-d) pie-d calcetín zapato-d zapato-i)

Figura 4.3 Secuencia que indica la acción "poner"

Cuando una persona observa atentamente la acción "poner" puede realizar el seguimiento del movimiento continuo de cada objeto. Sin embargo, en este caso, se suprime la descripción detallada de la secuencia, indicando explícitamente al sistema qué elementos se unieron y en qué se transformó la imagen.

En un sistema más avanzado sería importante incluir descripciones más detalladas de los objetos. Esto permitiría extraer la relación entre las imágenes anteriores y la nueva, tales como el hecho de que un calcetín sobre un pie tiene forma de pie pero con la textura del calcetín. Por el momento se tendrá que dejar solamente la indicación de que ha aparecido una imagen u objeto nuevo.

La notación básica aquí introducida, será utilizada para describir cada imagen integrante de las secuencias de acciones estudiadas durante el resto de este trabajo.

4.2 JERARQUÍAS

Hasta el momento, se ha trabajado en la representación de objetos individuales sin mayor relación entre ellos. Se presenta a continuación la propuesta de este trabajo para introducir un mayor manejo de información relativa a las imágenes observadas.

La secuencia completa de acciones puede ser descrita con la notación ya propuesta, sin embargo, aún falta incluir cierta información que se encuentra a la disposición de una persona y que es indispensable para la extracción de reglas. Esta información está relacionada con las similitudes entre los objetos: no es posible considerar que un pie derecho es tan distinto de un pie izquierdo como lo es de una mano. Esta relación queda establecida "nada más con mirar las formas", cosa que una persona hace automática e inconscientemente.

Anteriormente se mencionó que, para este problema, se asume que el sistema de visión del agente es capaz de transmitir la descripción de las imágenes explicitando algunas similitudes entre ellas. En una primera aproximación, esta información se incluye diciendo que una mano derecha es un tipo de mano, al igual que lo es la mano izquierda; similarmente se hace para los pies y los zapatos. Esto es, claramente, representable por el concepto de "clases" y "herencia de clases", del paradigma orientado a objetos. Se asume, por lo tanto, que el sistema de visión es capaz de manejar una jerarquía de clases en la cual cataloga las imágenes que percibe. La definición de una relación "clase - clase(s) hija(s)" será utilizada por los algoritmos planteados más adelante.

Esta relación entre imágenes se puede denotar utilizando el símbolo => de la manera siguiente:

```
(Clase-padre => imagen-hija) ó ((lista-clases-padres) => imagen-hija)
Ejemplo: (mano => mano-derecha)
```

Es importante remarcar que el nombre del objeto no se está utilizando más que para distinguir a una imagen de otra.

Tratándose de un catálogo de imágenes, se dice que las clases ingresadas son "clases de imágenes". Los nombres que se utilizan en las descripciones, son en realidad, el nombre de la clase a la cual pertenece el objeto identificado. Al igual que ocurre en el paradigma orientado a objetos, pueden existir varias instancias de una misma clase; se abundará en este punto más adelante. Además, en esta aplicación, no es necesario que todas las clases desciendan de alguna otra y una clase puede tener tantos ancestros como se considere conveniente.

La figura 4.4 muestra, en forma gráfica, cómo se describiría la posición que ocupa la imagen V3 en la jerarquía de clases ilustrada. Esta imagen da un ejemplo del uso de herencia múltiple.

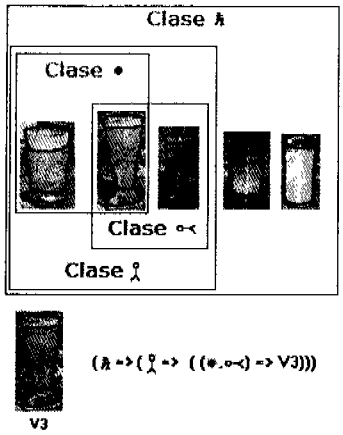


Figura 4.4 Notación utilizada para representar a todos los ancestros del vaso con malteada tipo V3. Se puede observar que, el vaso mencionado pertenece a dos clases en el primer nivel: vasos con contenido color claro y vasos para malteada. En el segundo nivel pertenece a una categoría de vasos con contenido espumoso (recordar que son catálogos por las propiedades de las imágenes, no por lo que en verdad son los objetos). Por último, pertenece a la categoría de recipientes casi cilíndricos.

Esta jerarquía de imágenes es uno de los puntos que podrían ser de utilidad para obtener un concepto más completo del objeto con el que se está tratando.

La notación para el estado inicial del problema base, ya incluyendo estos elementos, es una lista de los objetos visibles al inicio de la acción: las dos manos, los dos pies, dos calcetines y los dos zapatos. Esto es:

```
((mano => mano-izquierda) (mano => mano-derecha) (pie => pie-derecho) (pie =>
pie-izquierdo) calcetín calcetín (zapato => zapato-d) (zapato => zapato-i))
```

Por brevedad, lo anterior se escribe:

```
((mano => mi) (mano => md) (pie => pd) (pie => pi) cal cal (zapato => zd)
(zapato => zi))
```

Después de haber especificado el tipo general de cada imagen ya no es necesario volverlo a escribir. Internamente, el agente se va encargando de guardar una jerarquía en la cual se encuentran ubicadas todas aquellas imágenes-objetos distintas que ha "percibido". Esta jerarquía de clases es almacenada como una gráfica dirigida y cada nodo hijo es una clase más específica que su padre.

La figura siguiente muestra el bosque que se genera tras la lectura de la primera imagen.

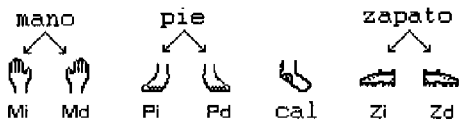


Figura 4.5 Jerarquía de clases generada a partir de la lectura de la descripción de la primera imagen.

Claramente, el tipo de relación establecida entre la clase más general y las más específicas es una relación de herencia.

Internamente, el agente mantiene una tabla con acceso directo a cada una de las clases registradas (padres incluidos). En una implementación de un sistema verdaderamente grande no toda la tabla debe estar cargada en memoria todo el tiempo. Además, cada clase contiene ligas a sus padres y a sus hijos.

4.2.1 Grupos

La representación interna del medio en que se debe desenvolver un agente incluye una descripción espacial. En este caso, dicha descripción espacial se limita a la formación de grupos de objetos que se encuentran en contacto físico. Cada uno de estos grupos también tiene asociado un tipo. Este tipo es la clase del grupo.

El tipo de un grupo es el nombre asociado a un nodo en la gráfica de clases, que apunta hacia los tipos de todos los elementos que contiene. Se dice que los objetos contenidos en el grupo son atributos suyos.

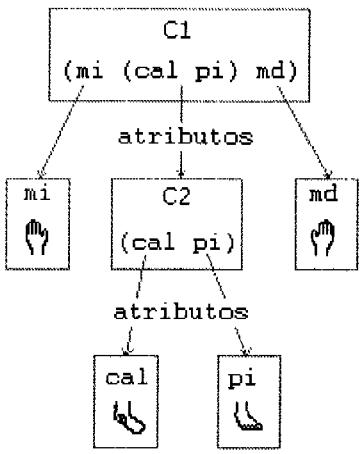


Figura 4.6 Un grupo de objetos tiene asociado un nodo con apuntadores a los tipos de sus integrantes.

En una primera aproximación, en caso de que algún grupo contenga más de un elemento del mismo tipo, deberá haber tantas ligas entre éste y el tipo de los elementos repetidos, como elementos repetidos contenga. Es posible elegir una representación más compleja que, en lugar de aumentar el número de ligas, lleve un conteo de los elementos de cada tipo.

El uso de un identificador único para cada grupo de objetos de distinto tipo permite hacer un símil de un "concepto global". Por ejemplo, un ser humano puede distinguir a una persona tan sólo con ver su silueta. No es necesario entrar en los detalles acerca de los ojos, tipo de nariz o color de la piel, para asociar una silueta de persona con el concepto correspondiente. Los detalles se ven después, para identificar si se trata de alguien conocido.

De este mismo modo, una vez reconocida una imagen, este sistema puede manipularla usando su tipo sin tener que recurrir siempre a sus partes. Pero, si requiere conocer las partes, el nombre de la clase (tipo) le permite acceder al nodo correspondiente y, desde éste, a sus atributos.

Se podría pensar que esto provocaría una diferenciación demasiado marcada entre grupos como: las manos sosteniendo al calcetín y las manos sosteniendo al pie con el calcetín. Este caso se ilustra en la figura 4.7. Este problema será subsanado gracias al criterio utilizado para el manejo de la herencia con grupos.

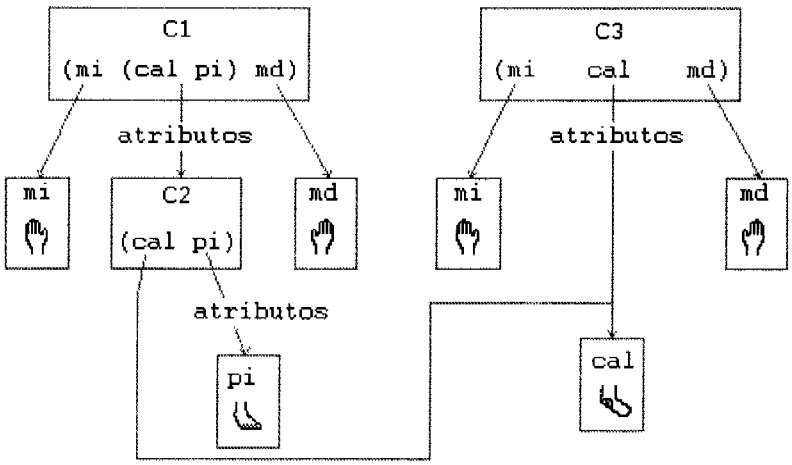


Figura 4.7 Dos grupos similares reciben nombres distintos.

Aquellos grupos con elementos en común heredan de una misma clase que contiene ligas a los atributos comunes. Los nodos hijos contienen las ligas a los elementos diferentes.

La figura 4.8 muestra cómo se reacomodan los elementos de las clases/grupos de la figura 4.7, para reflejar qué atributos son comunes a C1 y C3. Esta reestructuración corresponde a la aparición de un nuevo nodo padre (C4 en la figura), que contiene los elementos comunes, tal y como indica el paradigma utilizado, evitando la redundancia en el almacenamiento de la información y estableciendo una relación de similitud entre ambos grupos.

Es de suma importancia destacar que la definición original de C1 y C3 nunca se altera al introducir C4. Lo único que se modifica es la representación interna, pero esto es transparente para cualquier llamada al método que devuelve los atributos que conforman a alguno de los hijos. Acordemente a la definición de herencia, dicho método deberá devolver los atributos en la clase C4 (padre) más los específicos del hijo.

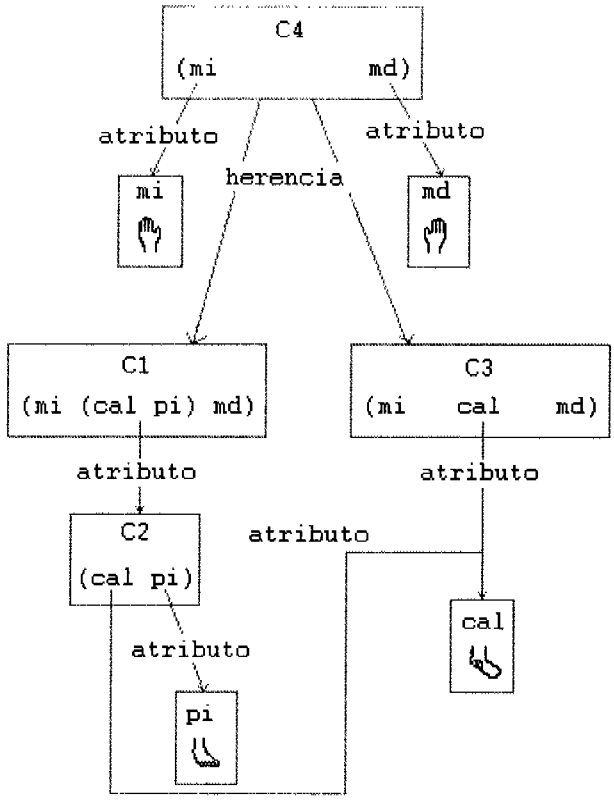


Figura 4.8 Una clase padre contiene los atributos comunes a varios grupos.

En la práctica, esta estructura facilita encontrar grupos con elementos en común o muy similares. Además, esta forma de reestructurar la descripción de los grupos pretende favorecer la agrupación de las ligas que apuntan a objetos, que comúnmente aparecen juntos, dentro de un mismo nodo. Sin embargo, dado que esta jerarquía es generada conforme el agente va percibiendo los objetos, agregar y redistribuir nodos puede convertirse en un problema muy complejo.

Es importante destacar desde un principio que, con una representación de clases tan sencilla, no es posible incluir demasiada información acerca de los objetos. En el caso de los grupos, únicamente se pueden listar sus componentes, agregar cualquier otra cosa daría origen a interpretaciones equívocas. Por ejemplo, si se intentara decir que un zapato es negro, agregar "negro" como atributo suyo, haría parecer que se trata de un grupo que incluye al elemento "negro".

En realidad, la notación original de clases agregaría los atributos dentro de rendijas con nombres. En el ejemplo, se agregaría "color: negro" o "elementos: (pie calcetín)". Sin embargo, será necesario restringir, por el momento, el tipo de atributos que puede tener cada clase, dados todos los procesos que se desea cubrir en el presente trabajo y que se volverán más complejos al agregar distintos tipos de rendijas (tipos que también se desearía agregar dinámicamente).

4.2.2 Generación de la Jerarquía

Inicialmente, el sistema no cuenta con ningún nodo en la jerarquía de clases. Éstos van siendo agregados dinámicamente conforme van siendo presentados por el sistema de visión.

Para aquellas imágenes que contienen exclusivamente elementos atómicos el proceso de asignar y agregar tipos resulta extremadamente sencillo. El procedimiento buscar/agregar para átomos es el que sigue:

- Buscar el nombre del átomo en la tabla de clases.
- Si no existe:
  - Agregar el tipo a la tabla.
  - Si el tipo descrito incluye información acerca de sus padres:
    - Buscar y/o agregar a sus padres.
- Devolver el tipo.

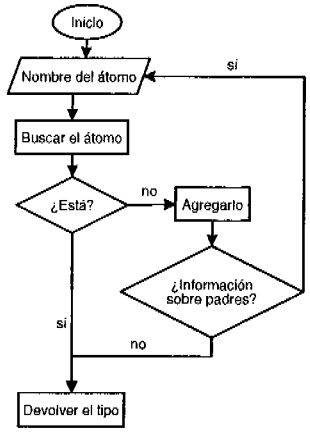


Diagrama 1. Diagrama de flujo del algoritmo buscar/agregar para átomos simples.

Dada la notación elegida para describir el contenido de cada imagen, no es posible que el sistema identifique inmediatamente la silueta de cada grupo. Por el contrario, se hace necesario que identifique al grupo completo como la unión de sus partes. Una mejor representación podría incluir información acerca de la silueta global, antes de hacer necesario entrar en detalles. Esta notación extendida tendrá que ser diseñada para aprovechar las características del problema de visión, sin embargo se deja como una extensión a futuro.

A este nivel, el sistema solamente puede distinguir entre átomos y grupos (imágenes simples y complejas). En caso de encontrarse con la descripción de un grupo, el algoritmo buscar/agregar es más complejo.

En primer lugar, la detección del tipo tendrá que ser realizada nivel por nivel, pues es posible tener grupos de grupos y no será posible adquirir el tipo de un grupo sino hasta que todos sus miembros internos hayan sido identificados. En segundo lugar, en caso de que sea requerido agregar un nuevo tipo para el grupo descrito, será importante seleccionar el lugar adecuado en la gráfica para el nuevo tipo.

La representación de los grupos expuesta anteriormente es un árbol. Para detectar el tipo de un grupo, conviene realizar un recorrido post-orden, de modo que cada nodo padre vaya acumulando los tipos de sus hijos. Cuando los tipos de todos los hijos estén dados, será posible identificar el tipo del padre, como se muestra en la figura 4.9.

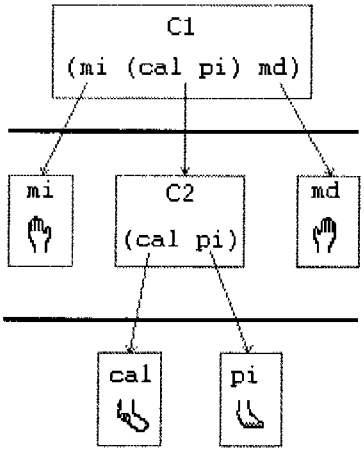


Figura 4.9 Para detectar el tipo de un grupo es necesario tener el tipo de cada uno de sus integrantes, los cuales pueden ser grupos también.

Encontrar el tipo de un nodo-grupo requiere de una búsqueda: dados los tipos de los atributos, encontrar al grupo que los contiene a todos ellos. En caso de no haber ningún nodo que satisfaga esta condición, se desea obtener aquel primer nodo que posea más atributos en común con el grupo nuevo. Este nodo sólo será útil si los atributos compartidos son al menos la mitad de los atributos poseídos por dicho nodo. Esta restricción favorece que solamente queden expresadas relaciones significativas entre los grupos.

A partir del nodo seleccionado, la jerarquía de clases debe ser reestructurada para agregar al nodo nuevo de acuerdo al algoritmo siguiente:

- Si el nodo encontrado (E) está completamente contenido en el grupo nuevo (N): E se convierte en el padre de N, que sólo deberá agregar ligas a sus atributos faltantes.
- Si N está completamente contenido en E, N se convierte en el padre, con ligas a los atributos comunes y E es reestructurado, eliminando las ligas a los elementos que contiene N.
- Si E y N solamente comparten algunos atributos, se realiza la siguiente reestructuración:
  - Se crea un nodo nuevo (C) con los atributos compartidos por E y N.
  - E y N se vuelven hijos de C.
  - N agrega ligas a los atributos faltantes.
  - E elimina sus ligas a los atributos contenidos en C.

Aparentemente el algoritmo anterior produciría una jerarquía con las características deseadas. Sin embargo, si se toma en cuenta que el nodo E

puede ya estar descrito como heredero de otras clases, aparecen varios casos cuyo tratamiento no es evidente. Uno de ellos se muestra en la figura siguiente 4.10.

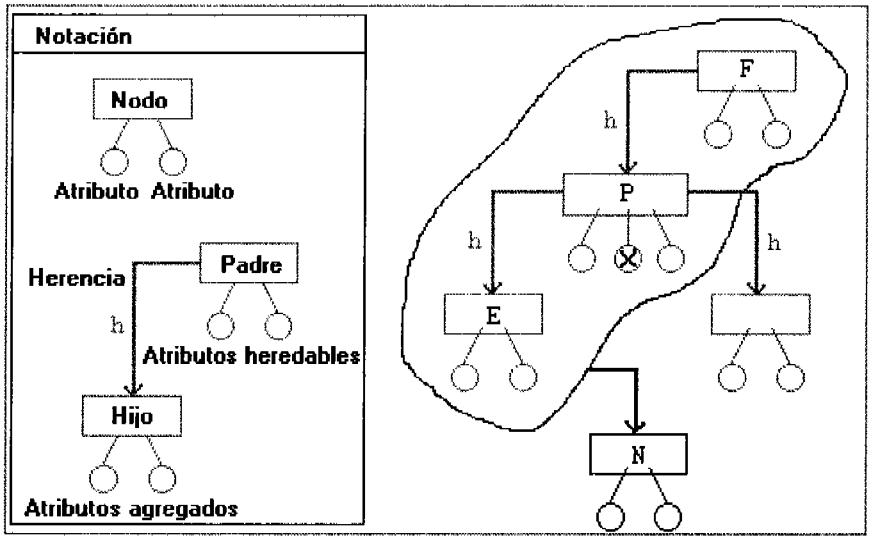


Figura 4.10 El grupo representado por el nodo E, hereda atributos de los nodos P y F. El nuevo grupo (N) comparte casi todos sus atributos con el grupo E, excepto uno que pertenece a un ancestro (marcado con una x) y dos atributos nuevos, que debe agregar a su descripción.

En la figura se muestra un caso en que el grupo nuevo (N) comparte todos sus elementos con un grupo en la gráfica (E), excepto por uno, que el grupo nuevo no contiene, y otros dos que debe agregar. De acuerdo al algoritmo descrito, debiera crearse una clase padre para estos dos nodos, con los nodos comunes. El problema es que el atributo que no se comparte con E, pertenece a una clase padre de E. Observando el problema a detalle, no resulta obvio como evitar incluir ese atributo. Ese atributo debe pertenecer tanto a E como a todos los hijos de P. Al mismo tiempo, se desea que N tenga los atributos agrupados en el nodo E, pero no el que heredó de P.

Podría parecer, entonces, que lo que se desea es que la descripción de N sea la unión de los nodos E y sus ancestros, menos el atributo no deseado. Pero entonces las relaciones de herencia ya no podrían ser utilizadas ni aprovechadas de la misma manera. Las similitudes entre E y sus hermanos, hijos de P, se perderían.

Existe una opción que permitiría conservar el parentesco a través de P y la nueva relación con N. Esta opción consiste en utilizar un grafo Y-O. La combinación de Y-O con herencia es un planteamiento nuevo, especialmente en este contexto. Este tipo de descripción, como puede verse en la figura siguiente, no es un caso de herencia múltiple.

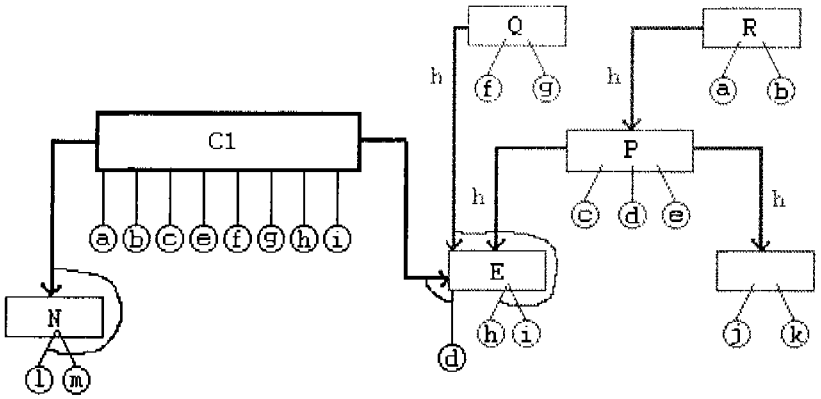


Figura 4.11 En la figura se ilustra el uso de un grafo Y-O combinado con herencia. Los arcos entre las ligas de atributos y de herencia indican que estos elementos juntos dan la definición del nodo. Esto se muestra en los nodos **N** y **E**. **N** hereda los atributos **a**, **b**, **c**, **e**, **f**, **g**, **h** e **i** de **C1** y agrega **l** y **m**. **E** corresponde a un caso en el que un mismo nodo puede ser descrito de dos formas distintas, las cuales quedan separadas por la ausencia de arco; como que **E** hereda de **C1** y agrega **d** o que hereda de **Q** y **P** (el cual a su vez hereda de **R**) y agrega **h** e **i**. **E** también muestra la notación para herencia múltiple: el mismo arco abarca ambas relaciones de herencia y los atributos que se agregan. Los arcos se omitieron para el resto de los nodos para simplificar la figura.

En una representación que combine herencia y grafos Y-O se están almacenando distintas formas de agrupar los elementos contenidos en un mismo nodo. La gran desventaja radica en que la cantidad de diferentes representaciones para un mismo nodo puede ser extremadamente grande. Definitivamente no vale la pena tenerlas todas, por lo que habría que agregar criterios para seleccionar cuales se quedan.

Hay varias características que pueden influir al seleccionar qué representaciones utilizar:

- Aquellas que aparezcan con mayor frecuencia.
- Aquellas que maximicen las similitudes entre los objetos emparentados.
- Aquellas que sean más útiles en los razonamientos subsiguientes, es decir, las que permitan encontrar patrones con mayor facilidad.
- Aquellas en concordancia con los conceptos previamente establecidos por otros agentes. Es decir aquellas estructuras que hayan sido prefijadas por otros entes en una sociedad y que pueden servir como puntos de referencia para la comunicación.

En general, ya sea utilizando grafos Y-O o eligiendo algún otro criterio para reestructurar la jerarquía de clases ante la aparición de elementos nuevos, se trata de mantener una estructura para la representación interna que pueda ser utilizada en la definición de conceptos. Un concepto surgirá de las relaciones que se puedan identificar entre los diferentes elementos memorizados.

Mientras que la estructura de herencia permite encontrar semejanzas entre un número limitado de elementos emparentados, bajo un solo criterio (la jerarquía que permanece); la inclusión de grafos Y-O permite agrupar elementos bajo diversos criterios de semejanza (guardando jerarquías

alternas), con el costo de ocupar más espacio y acrecentar el dominio de las búsquedas.

Cualquiera de los criterios anteriores puede llevar a elecciones distintas dependiendo de las circunstancias. Por el momento, para el problema a desarrollar aquí, basta con utilizar el algoritmo tal y como quedó descrito en primera instancia, pues los casos más complejos no se presentarán.

Éste es el primer punto donde se hace evidente un problema mayor que debe ser estudiado en detalle antes de construir un agente con capacidades generales de aprendizaje. Lo que se pone de manifiesto en las gráficas anteriores es que hay muchas formas de organizar los conceptos. La posibilidad de agregar conocimiento nuevo depende, en gran medida, de la estructura almacenada hasta el momento. De hecho, este es uno de los problemas que enfrenta el educador al tratar de inculcar conocimiento nuevo en sus pupilos: debe agregarlo sobre lo que los alumnos ya sabían de antemano.

En la práctica, algunas formas de organización de los conceptos pueden ser más prácticas que otras. Los esfuerzos de generaciones de investigadores por encontrar las clasificaciones más adecuadas corresponden a una ciencia: la taxonomía. Esta ciencia se aplica en particular, dentro de la biología, para la ordenación jerarquizada y sistemática, con sus nombres, de los grupos de seres vivos.

Las clasificaciones realizadas constituyen información que debe quedar a disposición del sistema independientemente de la situación concreta a partir de la cual fue generada.

4.3 El Experimento

A continuación, la lista denominada "guión-zapatos-1" describe una secuencia de imágenes en la cual se ve cómo los calcetines son colocados sobre los pies y luego, los zapatos sobre los pies con calcetines. Cada elemento en la secuencia se considera un cuadro del guión. Es importante recordar que estos "cuadros", no coinciden exactamente con "cuadros" de video.

```
LISTA guión-zapatos-1:

Elementos iniciales, son los miembros de un actor que se dispone a calzarse y el
calzado:
((mano => md) (mano => mi) (pie => pd) (pie => pi) cal cal (zapato => zd)
 (zapato => zi))

El actor toma un calcetín con ambas manos:
((md cal mi) pd pi cal zd zi)

Lo coloca en el pie izquierdo:
((md (cal pi) mi) pd cal zd zi)

El calcetín ya está puesto:
((md ((cal pi) -> (cp => cpi)) mi) pd cal zd zi)

Suelta el pie con el calcetín:
(md mi pd cal zd zi cpi)
```

```
Toma el otro calcetín:
((md cal mi) pd cal zd cpi)

Lo coloca en el pie derecho:
((md (cal pd) mi) cpi zd zi)

Puesto:
((md ((cal pd) -> (cp => cpd)) mi) cpi zd zi)

Suelta el pie con el calcetín:
(md mi cpi cpd zd zi)

Toma el zapato izquierdo:
((md zi mi) cpi cpd zd)

Lo coloca en el pie izquierdo:
((md (zi cpi) mi) cpd zd)

Puesto:
((md ((zi cpi) -> (zcp => zcpi)) mi) cpd zd)

Suelta el pie con el zapato:
(md mi cpd zcpi zd)

Toma el zapato derecho:
((md zd mi) cpd zcpi)

Lo coloca en el pie derecho:
((md (zd cpd) mi) zcpi)

Puesto;
((md ((zd cpd) -> (zcp => zcpd)) mi) zcpi)

Suelta el pie con el zapato:
'(md mi zcpd zcpi)
```

Cabe destacar que, al meter un objeto dentro de otro (como el pie, dentro del calcetín), el nombre de la imagen cambia. Este recurso permite simular el hecho de que "se ve" cómo las imágenes son modificadas al meter un objeto dentro de otro. Para indicar esta operación, se utiliza la notación:

```
((zd cpd) -> zcpd)
```

Esto indica que los elementos "zd" y "cpd" se han unido para originar a "zcpd". Esta operación será referida como "fusión". Una fusión solamente puede realizarse de un grupo a un elemento atómico, como se explicó anteriormente.

Un mejor sistema para describir las imágenes debería incluir información acerca del hecho de que un calcetín sobre un pie continúa teniendo textura de calcetín, pero adopta la forma del pie. El seguimiento que dé origen a estos cuadros descritos en forma simbólica, podría ser realizado utilizando Cadenas de Markov[24]. Una cadena de Markov es una serie de eventos, en la cual la probabilidad de que ocurra un evento depende del evento inmediato anterior. Para identificar patrones, se usan los "Modelos Ocultos de Markov". En ellos, se tienen observaciones que dependen de los estados, pero los estados son desconocidos. Se utilizan algoritmos como el de Viterbi para, dadas las observaciones, obtener el estado oculto más probable. Para reconocer, por ejemplo, objetos en imágenes, se deben extraer parámetros característicos de las imágenes mediante procesamiento digital y, dados éstos, se trata de identificar de qué objeto se trata (estado oculto), con mayor probabilidad.

Las Cadenas de Markov permiten identificar secuencias de estados, de tal forma que el seguimiento de las acciones resulta sensible a las modificaciones paulatinas de las formas y texturas visibles a través del tiempo. Juan María Sánchez [25] utiliza Cadenas de Markov para identificar segmentos de video que son semánticamente significativos. Para los alcances del presente proyecto, se omite el manejo de esta información.

Una implementación del algoritmo básico descrito, lleva a la generación de la jerarquía de clases ilustrada por la gráfica siguiente.

La figura 4.12 muestra varias de las propiedades que se buscaba en la creación de la jerarquía. Para leer esta gráfica es conveniente recordar que los atributos de cada grupo son aquellos que están ligados a la clase directamente, más aquellos que hereda de sus ancestros. Por ejemplo, los atributos de C#15 son "pi", que contiene directamente, y "cal", que hereda de C#18. De este modo, C#15 representa al grupo (cal pi).

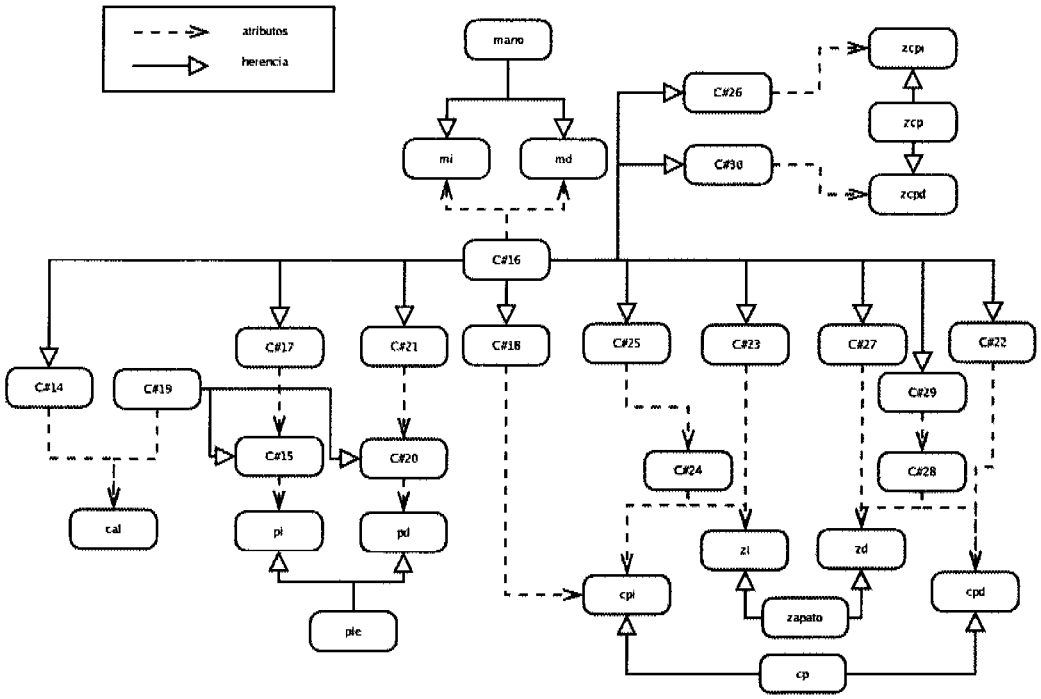


Figura 4.12 La jerarquía de clases generada automáticamente para los objetos y grupos participantes en una secuencia que describe cómo son puestos los calcetines y los zapatos. Un listado detallado del significado de cada símbolo se incluye en el Apéndice B.

Como se observa en el guión, ambas manos son utilizadas cada vez que se agarra un objeto. Como resultado, el sistema ha generado una clase que contiene como atributos a las dos manos (C#16). De esta clase heredan todas aquellas clases que corresponden a grupos que incluyen ambas manos y algún otro objeto.

Además, una persona sabe que, al meter un pie dentro de un calcetín, los objetos originales no desaparecen. Sin embargo, al nivel de visión, sus

imágenes sí lo hacen; tal vez totalmente o tal vez sólo en parte; pero ya no son iguales. Por ello, este hecho no se expresa en la descripción de la imagen.

Para ilustrar este caso, es interesante observar, nuevamente, el comportamiento de los infantes. Para un niño pequeño, puede ser todo un descubrimiento el observar que, tras meter una pelotita en un jarrón, ésta continúa ahí. La próxima vez que el niño observe cómo la pelota desaparece tras el borde del jarrón, recordará que la pelota permanece dentro. Algunos adultos disfrutan, entonces, simulando que meten la pelota, pero en realidad sólo la esconden detrás del jarrón. En este caso, ahora el niño se sorprende al buscar la pelota dentro del jarrón y no encontrarla. Ahora, intentará encontrar alguna manera de distinguir un caso de otro.

CAPÍTULO V

INSTANCIAS INTERNAS

5.1 LA REPRESENTACIÓN INTERNA

Una vez definida la notación que representa la información introducida por un sistema de visión, inicia el análisis de los procesos que se llevan a cabo internamente. Estos procesos permiten interpretar la sucesión de imágenes como una secuencia de transformaciones en el mundo. Un mundo que debe tener un símil en el interior de la mente del agente.

Ahora, para interpretar los cambios entre imágenes sucesivas, el agente debe asociar una instancia interna por cada imagen que ha recibido. Cada una de estas instancias se referirá a un único objeto. Es importante observar, en este punto, un hecho que todo ser humano asume inconscientemente: que cada objeto, tras haberse movido (cambiado de posición), sigue siendo el mismo.

Tal vez esta certeza viene de la sensación de cada quien, de seguir siendo el mismo aún después de haberse movido. En cierta forma, se puede pensar que cada individuo se experimenta a sí mismo como el único protagonista de una historia centrada en él. Esta sensación se extrapola automáticamente a todo aquello que ve a su alrededor: otras personas, otros objetos.

Inicialmente, una persona observa un movimiento continuo y sabe que es un solo objeto el que **se está moviendo**. Posteriormente, incluso si se trata de fotografías de algo conocido, sabe que son imágenes del **mismo** objeto que ya había visto antes. Incluso en las fantasías de ciencia ficción, en las que un personaje se teletransporta de un punto a otro, el observador sabe que se trata del mismo individuo.

Sin embargo, cuando una computadora analiza dos imágenes sucesivas no asume, naturalmente, ninguna conexión entre los objetos en cada una de ellas, aún cuando se vean "idénticos". Inicialmente, para poder realizar un seguimiento, es necesario incluir en el sistema una premisa: "dos objetos *suficientemente semejantes* en imágenes *sucesivas* y en posiciones *cercanas* **son el mismo** (... a menos que algo indique explícitamente lo contrario)". Ésta es la idea del movimiento continuo.

Entonces, analizar el movimiento requiere de dos acciones iniciales:

- A partir de la primera imagen en que aparezca un objeto, crear una instancia interna asociada a esa imagen.
- Para las siguientes imágenes, asociar la misma instancia a imágenes semejantes, dando preferencia, siempre, a asociar la misma instancia a la imagen semejante más cercana. Este punto es particularmente

importante para el caso de objetos no distinguibles, es decir, objetos que lucen idénticos pero que, de algún modo, se puede saber que **no** son el mismo (no son protagonistas de la misma historia).

Ahondando más en el último punto. No es difícil realizar el seguimiento para objetos que lucen esencialmente distintos de los demás pues, sin importar dónde se encuentren, siempre se puede interpretar que son el mismo, en primera instancia; pero no ocurre lo mismo con objetos indistinguibles.

Por ejemplo, piénsese en los calcetines del rígido mundo del agente: en principio es imposible distinguir a un calcetín de otro, sin embargo, es claro que se trata de dos objetos, por lo que, internamente, cada uno debe tener su propio identificador.

Cuando el sistema recibe la descripción de cada imagen, no tiene manera de adivinar cuántas siluetas iguales incluirá la descripción (como la mesa arreglada con varios vasos iguales). Además, el sistema no sabe qué es cada uno de esos objetos. Por esta razón, comienza por asignar un índice único a cada objeto que va citando la descripción. La figura 5.1 ilustra la asignación de identificadores. El sistema va leyendo la lista de imágenes de objetos, asigna un índice 0 a la primer ocurrencia de cada tipo de imagen e incrementa el contador en presencia de instancias repetidas.

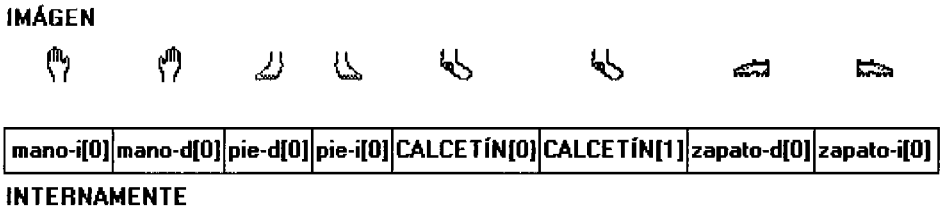


Figura 5.1 En la representación se asigna un identificador único a cada instancia.

Cuando las manos tomen uno de los calcetines, alguna de las instancias internas deberá ser asociada al calcetín entre las manos y la otra al calcetín que permanece separado. Hasta aquí, resulta indistinto cuál instancia es asociada a qué, (especialmente porque no se está tomando en cuenta la posición exacta de los objetos, sólo a qué grupo pertenecen); pero para el paso siguiente, ya no es lo mismo:

Uno esperaría que la instancia correspondiente al calcetín entre las manos, sea la misma que la asociada al calcetín que entra en contacto con el pie; mientras que el calcetín que ha permanecido fuera de la acción, debería tener asignada siempre la misma instancia.

La figura 5.2 ilustra cómo podrían ser asignados los índices, para este caso particular. Se asigna alguna de las instancias de calcetín al grupo recién formado por las manos, ésta puede ser la instancia 0 o la 1. Para el siguiente paso, la misma instancia elegida es la que debe ser asignada al nuevo grupo "pie izquierdo - calcetín".

IMÁGEN

mano-i[0]	CALCETÍN[0]	mano-d[0]	pie-d[0]	pie-i[0]	CALCETÍN[1]	zapato-d[0]	zapato-i[0]

INTERNAMENTE

```
   Imagen:
   ((mano-i calcetín mano-d) pie-d pie-i calcetín zapato-d zapato-i)
   Internamente:
   ((mano-i[0] CALCETÍN[0] mano-d[0]) pie-d[0] pie-i[0] CALCETÍN[1] zapato-d[0]
zapato-i[0])
   Opcionalmente:
   ((mano-i[0] CALCETÍN[1] mano-d[0]) pie-d[0] pie-i[0] CALCETÍN[0] zapato-d[0]
zapato-i[0])

   Si se toma la primera representación, en el segundo paso se espera:
```

IMÁGEN

mano-i[0]	CALCETÍN[0]	pie-i[0]	mano-d[0]	pie-d[0]	CALCETÍN[1]	zapato-d[0]	zapato-i[0]

INTERNAMENTE

```
   Imagen:
   (((mano-i (calcetín pie-i) mano-d) pie-d calcetín zapato-d zapato-i)
   Internamente:
   ((mano-i[0] (CALCETÍN[0] pie-i[0]) mano-d[0]) pie-d[0] CALCETÍN[1] zapato-d[0]
zapato-i[0])
```

Figura 5.2 Se desea que el mismo calcetín que fue tomado con las manos, sea en el que se mete el pie.

Esta asignación asegura que la interpretación de las acciones, que realice el agente, sea que un calcetín permaneció en reposo, mientras que el otro fue tomado por las manos y puesto sobre el pie. No se desea de ningún modo, que se piense en un calcetín saltando de afuera hacia las manos e intercambiándose con el de afuera para que éste esté luego colocado en el pie.

Este seguimiento debe realizarse así explícitamente en el código del agente.

5.2 SEGUIMIENTO A TRAVÉS DEL TIEMPO

En esta sección se trata el análisis de la evolución del medio a través del seguimiento de los cambios en los objetos descritos por las imágenes y en sus relaciones entre ellos.

El análisis de cada imagen, en la representación elegida, queda resumido a los siguientes pasos:

- Leer la descripción elemento por elemento.
- Detectar el tipo de elementos atómicos y grupos agregando los tipos que hagan falta.
  - Cuando se detecta a un objeto simple, el sistema busca su tipo en la jerarquía almacenada hasta el momento, si no lo encuentra, lo agrega.
    - A continuación se le asocia una instancia del tipo correspondiente al objeto visualizado.
  - Cuando se detecta la descripción de un grupo, se identifica el tipo del grupo a partir de las posiciones de los átomos que lo constituyen. De ser necesario, la jerarquía de clases es ajustada para incluir descripciones nuevas.
    - A continuación, se debe asociar una instancia a cada uno de sus componentes para, finalmente, crear una instancia de la clase del grupo con ligas a todas ellas. Sin embargo, este proceso de asociación depende del flujo del tiempo.

Ya se ha descrito cómo realizar la clasificación de las imágenes, sin embargo, el sistema de visión aún debe enfrentar la dificultad impuesta por la asociación de instancias.

Se desea que el sistema sea capaz de asociar una identidad única a cada objeto que participa en los guiones a analizar. Para lograrlo, es necesario profundizar en el análisis del concepto de movimiento.

Un agente, con un sistema de visión, percibe su entorno a través de secuencias de imágenes. La velocidad con que estas imágenes son percibidas varía para cada ser, así como la calidad. Para los propósitos de este trabajo, es posible considerar estas percepciones como secuencias discretas de descripciones fotográficas del entorno. Como en una película, las acciones son percibidas cuadro por cuadro.

Para una máquina, las secuencias de cuadros de un video no representan nada. La interpretación de dichas secuencias viene al tratar de encontrar las relaciones existentes entre un cuadro y sus antecesores, en orden cronológico.

Para buscar estas relaciones es particularmente conveniente segmentar las imágenes, buscando patrones que permitan comprimir la información contenida en ella y expresarla con la menor cantidad posible de símbolos. El estudio de la segmentación y detección de patrones en imágenes ocupa a muchos científicos e ingenieros al día de hoy. Para este proyecto, este paso fue sustituido por la descripción de las imágenes, mediante las listas de símbolos atómicos que se expuso anteriormente.

El primer tipo de relaciones que se pueden encontrar, corresponde a la detección de segmentos similares o idénticos, que aparecen en dos o varias imágenes, tal vez consecutivas.

La siguiente relación emerge al analizar las posiciones de los segmentos similares encontrados. Si las posiciones de segmentos idénticos son las mismas, se puede interpretar como que pertenecen a los mismos "objetos".

Un "objeto" corresponde a una entidad, ya sea corporal o espiritual, natural o artificial, real o abstracta [19]. Cuando se trata de una entidad corporal, puede tener asociada una imagen, originada por su interacción con la luz del medio ambiente, que es percibida e interpretada por el agente. Un objeto debe tener una representación interna, que exista más allá de una sola imagen y hacia la cual confluyan todas las referencias a esa misma entidad.

El agente tendrá que valerse de cuanta información posea a su alcance, para identificar a cada entidad a partir de sus atributos, que son los que percibe a través de los sensores. En este caso, el agente solamente cuenta con dos tipos de datos: forma y grupo al que pertenece el objeto.

Por otro lado, si se detecta un mismo segmento (o uno muy similar) en dos cuadros sucesivos, pero en una posición distinta (dentro de otro grupo, en este caso), el cambio puede ser interpretado como un **movimiento** del objeto. De este modo, el concepto de movimiento es algo que maneja intrínsecamente el sistema. Éste analiza las secuencias e interpreta las modificaciones de las imágenes en cuadros consecutivos, como desplazamientos de los objetos que las originan.

Sin embargo, decidir y representar qué es lo que se ha movido, no es una tarea sencilla. En realidad, este problema ha sido ampliamente tratado y es conocido como "el problema del marco".

Si todos los segmentos de una imagen fuesen distintos, bastaría con detectar, para cada posición visible, qué nuevos segmentos aparecen, cuáles desaparecen y cuáles se han desplazado (desapareciendo de su posición original y reapareciendo en otra para el cuadro siguiente). Pero éste no es siempre el caso.

Cuando existen varios segmentos idénticos (o vistas de objetos que no pueden ser distinguidos tan sólo mediante información visual), es necesario agregar un criterio que indique qué segmento de la imagen siguiente debe ser asociado con cada objeto.

Por lo tanto, la existencia de objetos que se ven iguales hace imperativa la aparición de una restricción más. Ésta restricción es utilizada automática e inconscientemente por los seres vivos y puede ser considerada como una heurística más que como un hecho. Ésta asunción consiste en ligar la misma identidad a las imágenes semejantes que se encuentren más cercanas espacialmente, entre cuadros consecutivos.

Considerando lo anterior, el seguimiento de los objetos a través del tiempo se puede realizar bajo la filosofía siguiente:

> *Al pasar de una imagen a otra (de un momento a otro) se debe considerar que objetos que se ven iguales y se encuentran en posiciones máximamente cercanas a la posición de su análogo en el cuadro anterior, son los mismos, a menos que algo indique explícitamente lo contrario*[1].

Para representar todo esto en la computadora y realizar el seguimiento de los objetos que percibe el agente, se hará uso de la jerarquía de clases generada anteriormente.

5.2.1 Creación y Seguimiento de Instancias

El análisis de una secuencia de acciones comienza con la identificación de los objetos que aparecen en la primera imagen de la secuencia (primer cuadro).

En general, para realizar el seguimiento de las instancias se requiere tener a disposición el cuadro inmediatamente anterior y el más reciente. Sobre ellos, se pueden detectar cuatro operaciones elementales para cada posición:

- Aparecer: un elemento o grupo, que no estaba en esa posición, ahora es visible.
- Desaparecer: un elemento o grupo, ya no aparece en la posición donde estaba registrado.
- Fusionar: dos elementos se unen formando una imagen completamente nueva. Es un prerrequisito que los elementos que se han de fusionar, estén en contacto (formen un solo grupo) en el cuadro inmediato anterior.
- Fisionar: una sola imagen da origen a dos o más elementos, que aparecen agrupados.

Las dos últimas operaciones son descritas explícitamente con el texto, ya que la representación actual no permitiría detectarlas de otro modo. Éstas son las más complicadas de detectar para el sistema de visión, pues deberá

---

[1] Este enunciado es el resultado de concretar los puntos que se han ido introduciendo a lo largo de la discusión planteada en este capítulo. Es posible que algo semejante se utilice en técnicas de procesamiento digital de imágenes, aunque el autor no cuenta con ninguna referencia.

pasar de los cambios paulatinos en los segmentos que detecta, a reconocer la acción completa de haber "fusionado" o "fisionado" dichos segmentos.

En el caso del experimento con el que se trabaja aquí, por "posición" solamente puede entenderse la pertenencia o no pertenencia a un grupo, ya que no se cuenta con más información.

Es muy importante destacar que estas cuatro operaciones elementales corresponden a cambios en las percepciones de agente, cambios entre imágenes. Estas operaciones le permiten al agente "interpretar" lo que ha ocurrido entre un instante y otro, pero no son más que manipulaciones internas.

Una vez obtenidos los tipos de cada elemento de la lista que describe el cuadro actual, el siguiente paso consiste en asociar a cada uno la instancia correspondiente.

Dado que, por el momento, no se trabajará con memoria de largo plazo, la primera imagen siempre contendrá únicamente objetos nuevos. De este modo, el análisis comienza desde cero con el primer cuadro del guión. Cuando se analiza esta primera imagen, ninguna instancia ha sido creada ni hay punto de comparación. Se trata del caso más sencillo:

- Para el primer cuadro, se crean instancias para cada objeto elemental y grupo que haya sido detectado, en la forma descrita en la sección anterior.

Al analizar las imágenes siguientes, será necesario identificar cuáles de las instancias detectadas en el cuadro anterior, permanecen en el mismo estado y cuáles no.

La imagen completa se encuentra descrita por los tipos de los objetos en el primer nivel. No es necesario citar uno a uno los elementos constituyentes de los grupos. Utilizando la notación obtenida en el experimento del capítulo anterior se tiene:

```
((md (cal pi) mi) pd cal zd zi) ≡ (C#17 pd cal zd zi)
```

Pero, deben tenerse instancias, en la representación interna, de los grupos como tales, con todos sus elementos. Esto se ilustra mejor con la figura 5.3.

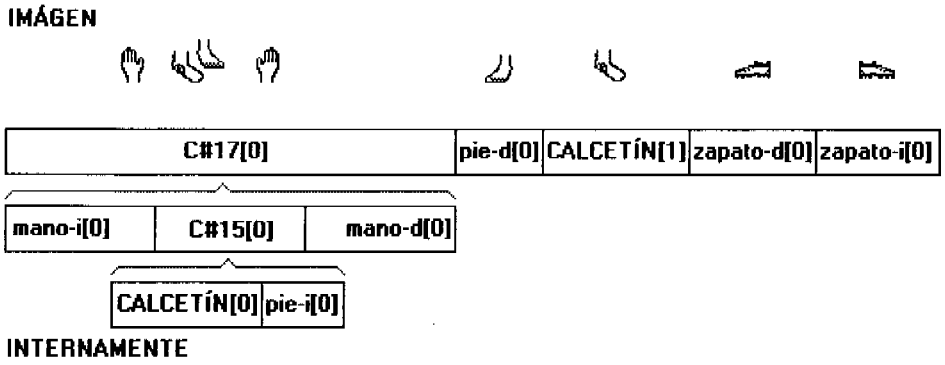


Figura 5.3 Internamente el sistema tiene referencias hacia las instancias de todos los objetos presentes en la escena, en una estructura que indica su posición física.

Gracias a esta forma de agrupar la información, no es necesario comparar los objetos, elemento atómico por elemento atómico, entre un cuadro y otro. Solamente es necesario profundizar cuando se detectan grupos distintos. Esto ocurre, por ejemplo, al pasar de tener sólo el calcetín entre las manos, a meterlo en el pie.

```
Cuadro i:      ((md cal mi) pd pi cal zd zi) ≡ (C#14 pd pi cal zd zi)
Cuadro i+1:    ((md (cal pi) mi) pd cal zd zi) ≡ (C#17 pd cal zd zi)
```

En el ejemplo anterior, de un cuadro a otro desaparece el pie izquierdo y el grupo C#14 ha sido modificado. Esto se refleja al identificar, en el cuadro siguiente, a C#17 en su lugar. Por supuesto, la distinción entre una desaparición de C#14 y aparición de C#17 y decir que C#14 se modificó a C#17 no es aparente. Para decidir cuál es la explicación más adecuada, es necesario analizar con más detalle qué relación puede existir entre C#17 y C#14.

En el caso de los átomos, es inmediato detectar si alguno ha aparecido o desaparecido, pero en el caso de los grupos es necesario analizar su estructura interna. Se requiere determinar qué grupo pudo haberse transformado en otro de los que se observan en la nueva imagen.

En primera instancia, el ejemplo anterior podría sugerir una comparación elemento por elemento de cada grupo. Sin embargo, piénsese en el caso en que haya más grupos. Realizar estas comparaciones para detectar cómo asociar los grupos modificados puede volverse demasiado complejo.

Dado que los movimientos son continuos en la mayoría de los casos, se espera que la mayoría de los elementos constituyentes de los grupos correspondientes sean iguales y que tengan estructuras muy parecidas. Por esta razón es posible utilizar la jerarquía de clases generada para viabilizar el análisis requerido.

Recuérdese que a cada grupo visualizado le fue asignada una clase que lo representa y por ello se distingue al grupo con el nombre de su clase. Estas

clases fueron acomodadas en la estructura jerárquica de acuerdo a las similitudes entre ellas. Aquí es donde se sacará ventaja de esta construcción.

En el caso del ejemplo anterior, la guía para saber qué grupo transformar en cuál, es que las clases C#14 y C#17 son hermanas. Ambas son hijas de C#16, la cual agrupa a ambas manos, mientras que sus hijas indican qué objeto es el que está siendo sostenido por ellas. De este modo, se espera que el grupo C#14 pueda ser transformado en C#17 mediante algunas operaciones sencillas, que incluyan el desplazamiento, muy probablemente, del elemento desaparecido en el primer nivel `(pi)`. (Y en efecto, así es).

Es importante recalcar que, a pesar de las hipótesis utilizadas, es probable que la asociación de instancias resulte arbitraria para algunos casos. Es el costo inherente a utilizar una heurística sencilla. Por ejemplo, al inicio, no hay forma de distinguir un calcetín de otro. Cuando esto ocurre no importa qué número de instancia se le asocia a qué objeto. Pero, cuando uno es tomado, se debe elegir alguna de las instancias y la elección queda "al gusto del agente". No hay ningún criterio que permita considerar una elección como mejor que la otra.

Algo semejante sucede cuando varios objetos aparecen y desaparecen reconformando los grupos visibles. Puede no ser obvio qué grupo fue transformado en cuál, especialmente si son muy parecidos y, por lo tanto, fueron almacenados como hermanos. Entonces, la interpretación de las variaciones entre cuadro y cuadro, variará dependiendo de la implementación del agente y la forma en que éste reciba la información.

Una vez conscientes de todo lo anterior, se puede discutir en detalle el algoritmo para seguir los cambios entre cuadro y cuadro.

5.2.2 El Algoritmo de Seguimiento de Instancias

El algoritmo que se expone a continuación recibe las descripciones de dos cuadros consecutivos. Sin embargo, la descripción del primer cuadro ya no se debe encontrar en términos de símbolos, sino de instancias. Como se ha explicado, el primer cuadro recibe instancias nuevas; a partir de este punto, se trata de determinar qué instancias permanecen en su lugar, cuáles se mueven, cuáles se han fusionado, cuáles desaparecen y cuáles aparecen, haciendo las asignaciones correspondientes.

Las comparaciones serán realizadas por niveles. La numeración de los niveles corresponde a la profundidad del árbol generado a partir de la descripción de la imagen. De este modo, la imagen global del medio ambiente corresponde al nivel 0. La figura 5.4 ilustra esta numeración con un ejemplo sencillo. (Los nombres de las clases/grupo no están relacionados con experimentos anteriores.) En el nivel 0 se lee la descripción de la imagen con los símbolos (C#0 `pi` C#1 `zi`); se dice que los elementos constituyentes de C#0 y C#1 pertenecen al nivel 1 y así sucesivamente.

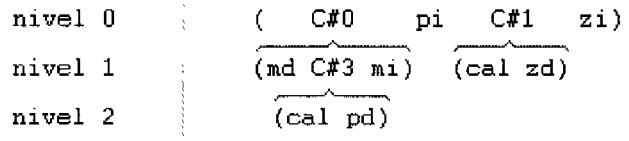


Figura 5.4 La descripción del medio ambiente se realiza con un árbol. Cada elemento atómico de la descripción corresponde a una hoja. La imagen global es el nivel 0.

Dado que el problema que está siendo analizado, no utiliza la operación de fisión, se ignorará por el momento.

A grosso modo, el algoritmo comparará elementos en el mismo nivel y será llamado recursivamente cuando sea necesario. En cada nivel se tendrán dos listas: una con las instancias asignadas a los objetos y/o grupos en el cuadro anterior y otra con los tipos de los objetos y/o grupos visibles en el cuadro siguiente. Por ejemplo, estas listas podrían verse así:

```
REPRESENTACIÓN INTERNA DEL CUADRO ANTERIOR:   (C#0[0] pi[0] C#1[0] zi[0])
DESCRIPCIÓN DEL CUADRO SIGUIENTE:                    (C#5 pi cal zd zi)
```

Lo primero que se hace es seleccionar aquellos símbolos que aparezcan en ambas listas. En el ejemplo: `pi` y `zi`. Para éstos simplemente se asigna la instancia de la lista anterior, a la imagen de la imagen siguiente. En el mundo descrito, esto corresponde a los objetos que no se han movido y ya no volverán a ser tomados en cuenta.

Para aquellos símbolos que no pueden ser mapeados directamente es necesario hacer análisis más profundo. Deben considerarse varios casos: que objetos hayan desaparecido de la escena; que hayan aparecido; que haya habido alguna reestructuración en el acomodo de los objetos por haberse cambiado de grupos o que se haya dado alguna fusión (lo cual sólo es reportado por la notación en el nivel en el que ocurre la fusión). Estos dos últimos se ven reflejados en los tipos de los grupos (los tipos son diferentes) y, en ocasiones, en la aparición o desaparición de elementos atómicos en diferentes niveles.

Precisamente la afectación de varios niveles es lo que más complica este análisis. El algoritmo debe llevar la cuenta de los objetos que han ido quedando sin asignar en los niveles anteriores y en las ramas por las cuales va descendiendo. El orden en el cual se haga el descenso también es importante.

Para resolver el primer problema, en la implementación se recurrió al uso de un repositorio que se pasa a cada llamada al método que implementa el algoritmo y es devuelto por éste ya con las modificaciones que realizó.

Para resolver el segundo se recurre, nuevamente, al conocimiento que se tiene del problema. Dado que se están describiendo movimientos continuos, es de esperarse que, a pesar de haber discretizado los cambios entre cuadro y cuadro, los grupos involucrados en los movimientos sean similares entre sí.

Nuevamente, la jerarquía de clases ayudará a determinar cuáles son los grupos semejantes, pues éstos deberán ser parientes cercanos. Lo que es más, dada la continuidad, si no aparecen y desaparecen objetos en la escena global, es aceptable asumir que siempre habrá un pariente o los integrantes separados (si el grupo se desbarató completamente).

El algoritmo deberá almacenar aquellos elementos atómicos sin pareja y emparejar a las clases que representan a grupos con sus parientes más cercanos (si es que tienen alguno). Esta elección pretende minimizar el número de apariciones y desapariciones reportadas, al transformar a la instancia anterior, en una correspondiente con la imagen nueva. El sub-algoritmo para determinar las parejas puede llegar a ser bastante complejo. Si se desea hacer un emparejamiento óptimo, se necesitará garantizar que todos han quedado con su pariente más cercano de entre el grupo a elegir. Por el momento, se manejó una implementación, no tan ambiciosa, que toma a las clases de la lista con menos elementos y, a cada elemento, le va buscando un pariente en la otra lista, en el orden que fueron guardados.

Una vez asignadas las parejas, el algoritmo es llamado recursivamente sobre éstas, hasta que ya no sea posible asignar más parejas de grupos.

En el ejemplo, los emparejamientos quedarían así:

```
ANTERIOR: C#0[0]        SIGUIENTE: C#5
```

Los demás son reportados como aparecidos o desaparecidos:

```
DESAPARECIDO: C#1[0]    APARECIDOS: cal zd
```

Este procedimiento se continúa aplicando hasta que, al final, se tenga una lista de asignaciones y una de elementos sin asociar. Inclusive, mientras se va descendiendo, el algoritmo también intenta asociar a los elementos sobrantes del nivel actual, con aquellos que estén esperando en el repositorio (elementos de otros niveles); buscando favorecer asociaciones entre niveles cercanos.

Al finalizar el recorrido se hace un último intento por asignar instancias a los elementos que quedan pendientes comparando los tipos de todos ellos. Para ello se recurre a desbaratar grupos que fueron reportados como desaparecidos, para encontrar instancias de sus átomos y solamente reportar la creación del grupo o su desbaratamiento, pero no apariciones y/o desapariciones de sus elementos. En el ejemplo, C#1[0] debe ser analizado en sus partes para darse cuenta de que sus elementos pueden ser asociados con `cal` y `zi` del nivel 0.

Aquellos elementos que hayan sido asociados exitosamente en pares instancia-imagen se reportan como *Movidos*, especificando de dónde a dónde se movieron. Los que hayan sido denotados como *Fusiones*, también tienen una asignación clara.

Aquellos elementos que restan son, efectivamente, elementos extra en las escenas y son reportados como tales, creándose y/o eliminándose las instancias correspondientes, de la representación interna.

Cada acción: *Mover, Fusión, Aparecer y Desaparecer* debe ser almacenada con todos los parámetros necesarios para actualizar la representación interna del estado del mundo. Es decir, de este análisis, el sistema debe aprender cómo modificar su representación interna del mundo (con instancias) al pasar de un cuadro a otro.

Una vez visto el funcionamiento general, a continuación se desarrolla el algoritmo *asigna-instancias*:

El nivel 0 es el punto de partida.

- Buscar aparecidos/desaparecidos:
  - Comparar, término a término, qué elementos (tipos) aparecen en ambas imágenes. Éstos ya no serán tomados en cuenta a partir de ahora.
  - Registrar las operaciones de *Fusión* y *Fisión* descritas en la imagen.
  - Para cada elemento que aparezca en el primer cuadro y no en el segundo, reportar *Desaparecer*, especificando la instancia y la posición del objeto.
  - Para cada elemento que no aparezca en el primer cuadro y en el segundo sí, reportar *Aparecer*, especificando el tipo y la posición del objeto.
  - Asociar a cada grupo desaparecido, un grupo aparecido (si hay más desaparecidos que aparecidos, si no la asociación se hace a la inversa); eligiendo para ello a su familiar más cercano. En caso de que no lo hubiera, se reporta que el grupo completo apareció o desapareció, según corresponda. Llamar *asigna-instancias*, recursivamente, con cada pareja asociada.
- Asociar, a cada elemento aparecido, la instancia del mismo tipo reportada como desaparecida de la posición más cercana.
  - Repetir mientras queden elementos aparecidos:
    - Si se encontró una instancia, *Mover* la instancia anterior a la dirección indicada por la imagen nueva. Guardar la acción *Mover*.
    - Si no se encontró nada, se verifica si no ocurrió una fusión.
      - Si ocurrió, ejecutar *Fusión*, actualizando la imagen del mundo. Guardar la acción *Fusión*.
      - Si no fue fusión:
        (a) Si quedan grupos reportados como aparecidos: Reportar la *Aparición* del grupo. Añadir la instancia para el grupo en el mundo y reportar a sus elementos como aparecidos. Guardar la acción *Aparición*.
        (b) Si no quedan grupos aparecidos, pero sí instancias de grupos reportadas como desparecidas: reportar la *Desaparición* del grupo, actualizar la representación del mundo y reportar a sus constituyentes como desaparecidos. Guardar la acción *Desaparición*.

- Si ninguno de los anteriores produjo un cambio, salir del ciclo.

- Si algún elemento aún está reportado como *Aparecido* y no como *Movido*, significa que el elemento no había sido visualizado en el cuadro anterior, por lo que se debe generar una instancia nueva. Aquellos que continúen reportados como *Desaparecidos* se eliminan de la representación del mundo. Actualizar la representación interna, para que refleje el nuevo estado del mundo y guardar el listado de las acciones requeridas para realizar esta actualización.

El algoritmo anterior permite almacenar únicamente las operaciones necesarias para transformar la representación interna del mundo correspondiente a la imagen anterior, para volverla acorde a la imagen actual. A raíz de esta reestructuración, los tipos de las instancias involucradas cambian.

Por lo anterior, la representación final de la acción completa (todas las modificaciones que ocurrieron de un cuadro a otro), debe solicitar al sistema que calcule dinámicamente los tipos de los grupos resultantes tras la aplicación de las operaciones de transformación (desplazamiento, creación y eliminación de instancias). Estos tipos deben corresponder con los observados en la imagen percibida a través de los sensores.

A partir de la información recabada por este algoritmo, se desea obtener una representación de las modificaciones detectadas entre cuadro y cuadro que sea independiente de la situación actual. Se desea generar una función nueva que:

- Reciba como parámetros los tipos de los objetos **participantes**, cómo localizarlos y los tipos de los objetos que se obtienen al aplicarla.

- Al ser ejecutada sobre la representación de cualquier mundo, que contenga los objetos necesarios, devuelva la representación del mundo actualizada.

Dado el algoritmo anterior, esta función puede obtenerse naturalmente utilizando un lenguaje de programación funcional. En el capítulo siguiente se expone en detalle el proceso de aprendizaje de acciones, que inicia a partir de este punto.

Una vez adquirida esta nueva función, el seguimiento de acciones se simplifica. La próxima vez que el agente observe que los tipos de los objetos modificados coinciden con los tipos de los argumentos y salidas de alguna función registrada, sólo debe llamarla. Ya no será necesario realizar el análisis detallado de las diferencias entre las imágenes para decidir cómo reorganizar las instancias internas. Bastará con realizar una revisión de los tipos de los objetos, en el nivel 0 de ambas imágenes para determinar qué

función llamar[2]. Las funciones pueden ser almacenadas en tablas hash cuyas llaves sean precisamente los tipos de las entradas y las salidas. La implementación de este experimento funciona así.

Siguiendo el ejemplo utilizado recientemente, la función obtenida podría actuar sobre cualquier representación del mundo que incluya instancias de los objetos C#0 `pi` C#1 `zi` en el nivel 0. La función selecciona una instancia del mundo por cada tipo requerido y aplica sobre ellas las transformaciones aprendidas durante la fase de análisis anterior. De este modo, un estado actual que conste de:

```
REPRESENTACIÓN INTERNA DEL CUADRO ANTERIOR:
        (foo[7] C#0[0] pi[0] C#1[3] zi[2] otro_foo[9] cal[1] cal[0])
    Con:        C#0 = (md[0] C#3[0] mi[0])
        C#3 = (cal[1] pd[0])
        C#1 = (cal[2] zd[2])
```

Sería transformado en:

```
REPRESENTACIÓN INTERNA DEL CUADRO SIGUIENTE:
        (foo[7] C#5[0] pi[0] cal[2] zd[2] zi[2] otro_foo[9] cal[1])
 Con:   C#5 = (md[0] cpd[0] mi[0])
```

Lo cual podría interpretarse como una habitación algo desordenada (con calcetines idénticos esparcidos por ahí y objetos desconocidos), en la que un calcetín cae del interior de un zapato, mientras un individuo se pone otro en el pie derecho. Por el momento, el sistema no separa estos dos hechos que ocurren simultáneamente, sin embargo, esto se resuelve guardando acciones separadas para objetos que no interactúan durante el cambio de cuadros.

Por último, es importante destacar que, tras el análisis realizado, la descripción de las acciones ya únicamente involucra a los objetos participantes. Esto permitirá obtener reglas al estilo de CLIPS, que especifican cómo pasar de un estado a otro, mencionando únicamente lo que cambia. Esta característica de CLIPS evita el problema del marco.

---

[2] Recuérdese que el tipo de un grupo es una descripción directa de los elementos que lo constituyen.

CAPÍTULO VI

APRENDIZAJE

6.1 EL LENGUAJE FUNCIONAL LISP

Dos características especiales del lenguaje de programación LISP, en la implementación CLISP, han sido explotadas para la implementación de este sistema:

- La habilidad para manipular funciones como cualquier otro objeto del lenguaje: crear instancias en tiempo de ejecución, almacenarlas en estructuras de datos, devolverlas como resultado de la ejecución de otras funciones, etc. Es decir, se usan funciones de orden alto.

- El alcance estático. En este tipo de alcance, el valor de una variable, en tiempo de ejecución, depende del lugar en que la función y la variable fueron definidas al momento de escribir el código. En contraposición al alcance dinámico, en el cual éste valor depende del orden en que se ejecutan las funciones.

Es con la combinación de estas dos características que se hace posible extender las capacidades de análisis del agente, de una forma sencilla y elegante.

La técnica utilizada involucra la utilización de las cerraduras [20] de las funciones, como una estructura de datos más. Dentro de estas estructuras se almacenan: el listado de objetos del estado inicial que tomarán parte en la acción, sus posiciones y las transformaciones que se deben hacer para llevar la representación interna al estado final. De este modo, el sistema aprende de los análisis que va realizando y, en presencia de la misma acción, ya no necesita realizar el análisis otra vez: sólo invoca la acción.

La idea consiste en utilizar una función que se encarga de generar funciones más particulares. Los datos que se entregan como parámetros a la función generadora, quedan dentro del alcance de la función generada. Dado que el lenguaje utiliza alcance estático, los valores de los argumentos, en el momento en que la función generadora es invocada, quedan almacenados dentro de la cerradura de la función generada.

De este modo, cuando la función generada es invocada, puede utilizar aquellos valores. Cada función generada guarda valores distintos para las variables.

El código siguiente, traducido del libro de Graham, ilustra este concepto de manera sencilla.

```
>(defun fábrica-sumador (n)              ;definición de la función fábrica
         #'(lambda (x) (+ x n)))

> (setq suma2 (fábrica-sumador 2)        ;función que suma 2 a x
        suma10 (fábrica-sumador 10))     ;función que suma 10 a x
> (funcall suma2 5)                      ;se usa la función suma2 con x=5
7
> (funcall suma10 3)                     ;se usa la función suma 10 con x=3
13
```

En este pequeño código, se define una función que fabrica sumadores, cada vez que es invocada, devuelve una nueva función, que suma el valor *n* al número que se pase como parámetro.

En este trabajo, esta técnica es utilizada para generar funciones que ejecutan las secuencias de transformaciones elegidas, sobre diferentes parámetros de entrada. Las secuencias, sus argumentos y salidas son pasados como parámetros a la función fábrica; ésta les agrega el código necesario para ejecutar la secuencia de acciones elementales sobre cualquier representación del mundo, que contenga los elementos necesarios para que la función sea aplicable en ella.

6.2 Aprendizaje de Acciones Cuadro por Cuadro

El último algoritmo expuesto en la sección anterior, además de actualizar la representación interna del mundo, llevaba el registro de las transformaciones básicas que va ejecutando. Concretamente, la información que guarda durante su ejecución son apuntadores a las funciones utilizadas (implementaciones generales de las funciones primitivas) y las direcciones de sus argumentos.

Dado el sistema de tipos definido, las dirección de cada elemento se indica mediante una lista que inicia con el tipo del objeto afectado en el nivel 0. Si se trata de un objeto dentro de un grupo, se inicia con el grupo que lo contiene en el nivel 0, el siguiente elemento de la lista es el tipo del elemento dentro de ese grupo... y así sucesivamente hasta llegar al elemento sobre el cual se realiza la transformación elemental.

Por ejemplo, utilizando la imagen de la figura 5.4, para referirse a `cal` dentro de C#0, la lista sería (C#0, C#3, `cal`).

Para seleccionar las instancias, se toma, de la representación del mundo, la primera instancia que tenga el tipo indicado por la lista de direcciones. Una vez dado este paso para el primer nivel, el sistema de tipos garantiza que será posible encontrar al objeto concreto, si se continúa descendiendo hasta encontrar el objeto solicitado en la profundidad requerida. Dado que dos instancias con el mismo tipo en el nivel 0 son indistinguibles, cualquiera que sea seleccionada es igualmente válida. Por eficiencia se toma siempre la primera encontrada.

Al analizar la acción realizada de un cuadro a otro, las funciones utilizadas incluyen funciones primitivas (*Aparecer, Desaparecer, Mover, Fusionar y Fisionar*[3]).

La función que se crea para reproducir las transformaciones, ya sobre cualquier representación interna del mundo (conjunto de instancias) sólo requiere que se le pase como parámetro dicha representación. La cerradura ya contiene la información acerca de qué tiene que buscar e incluye el código para seleccionar las instancias. Luego, invoca a las funciones en la secuencia indicada, para finalmente actualizar los tipos de las instancias afectadas.

Se identificará cada una de estas funciones como acciones-cuadro.

6.3 CARACTERIZACIÓN DE LA SECUENCIA

El mismo procedimiento se puede repetir sobre la secuencia completa, pero ahora, en vez de usar las funciones primitivas, la unidad más pequeña será cada acción-cuadro detectada. Es importante destacar la diferencia conceptual entre las funciones primitivas y las acciones-cuadro.

Las funciones primitivas fueron definidas para interpretar lo que el agente percibe a través de su sistema de visión. Su implementación se encuentra estrechamente relacionada con el tipo de representación interna del sistema pues actúan sobre ella. Son un criterio interno que, aunque pueda considerarse en cierto sentido, arbitrario, permiten al agente actualizar su representación interna del mundo, para irla haciendo acorde a las nuevas percepciones.

Por otro lado, si bien las acciones-cuadro también requieren ser descritas con la terminología de la representación interna, sí representan directamente las modificaciones que se han dado en el mundo, no las manipulaciones internas del agente. Estas acciones son la unidad más pequeña de movimiento, pues indican lo que ha ocurrido entre un cuadro y otro, que es "la medida de tiempo más pequeña para el agente". (Efectivamente, este agente tiene una percepción discreta del tiempo y cada intervalo es considerablemente grande).

Se espera que cada secuencia a analizar, corresponda a un procedimiento para alcanzar un objetivo. De este modo, la firma que identifica la secuencia se calcula comparando el estado del mundo al inicio y al final.

---

[3] La operación *fisionar* no fue implementada por no requerirse para este experimento, pero agregarla es inmediato.

Como argumentos, quedan registrados aquellos objetos (o grupos) que están presentes al inicio, pero no aparecen al final. En este experimento, son los pies descalzos, los calcetines y los zapatos.

Aquellos objetos que nunca fueron modificados (un florero en el escenario, por ejemplo) o que volvieron a su estado inicial, no son considerados como argumentos; en este ejemplo, es el caso de las manos. Sin embargo, aquellos que se requieren para realizar el proceso, son almacenados como "objetos auxiliares". De este modo, si se desea, el sistema podría verificar si cuenta con todos los elementos para ejecutar la secuencia completa, antes de empezar. Si no se hace la verificación, se obtendrá un error en el paso que requiera al objeto auxiliar. Nuevamente aquí entran ambas manos.

Las salidas son aquellos objetos que aparecen modificados al final. En este caso, lo que aparece son los pies ya con los calcetines y los zapatos.

Para este caso, la función *fábrica* regresa algo ligeramente distinto.

Cuando se analizó lo que ocurría entre cuadro y cuadro, el orden en que fueran realizadas las transformaciones era independiente de lo que ocurría efectivamente en el mundo exterior. Por esta razón, se almacenó la secuencia de transformaciones tal y como la obtuvo el agente. En cierta forma, es tan sólo un asunto interno. El punto importante era obtener una correspondencia entre las imágenes anterior y actual.

En el caso de la secuencia de acciones, sí es importante considerar el orden en que es posible realizar las acciones en el mundo real. De hecho, este es el punto que ha motivado esta tesis. Por esta razón, no se almacena la secuencia dentro de la cerradura de una función. Solamente se almacena una lista con las funciones-cuadro detectadas, en el orden en que fueron vistas.

Dada una función que pueda ejecutar estas secuencias, se puede reproducir la idea anterior: el sistema podría reproducir las transformaciones ya sin necesidad de presenciar los hechos. Incluso, se podría utilizar para realizar un seguimiento acción por acción, si el agente vuelve a presenciar el mismo hecho. El nuevo reto consiste en que, especialmente para este problema, podría darse que la secuencia no sea exactamente la misma. Entonces, el sistema debería ser capaz de darse cuenta de la diferencia; pero, al final, detectar que se está alcanzando el mismo objetivo.

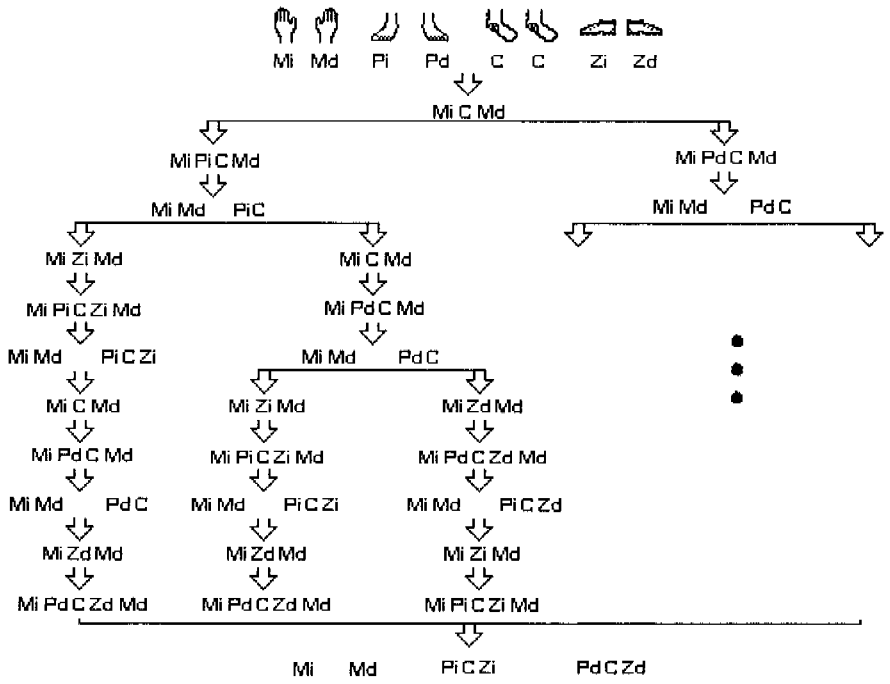


Figura 6.1 Existen seis secuencias distintas en las cuales se colocan los calcetines y los zapatos. NOTA: Por simplicidad no se listan todos los cuadros de la simulación.

Concretamente, en esta representación del mundo, existen seis secuencias distintas para resolver el mismo problema. Dadas las reglas correctas, CLIPS será capaz de recorrer este árbol y generar alguna de estas soluciones, dependiendo del orden en que realice la búsqueda. Para obtener dichas reglas, se deben tomar en cuenta todos los caminos posibles. (A menos que se quiera obtener el resultado más burdo, que consistiría en listar cada una de las acciones atómicas).

Por consiguiente, si se desea que el agente "profundice en su conocimiento acerca de la acción completa realizada", se puede dar aún un paso más.

En la figura 6.1 se han representado las alteraciones en la secuencia como apariciones de ramas en un árbol. Cada una de las bifurcaciones indica la posibilidad de ejecutar alguna otra acción; es decir, indica cuándo el orden a seguir ya no es estricto.

El último algoritmo importante, antes de traducir la información recopilada, al lenguaje de la máquina de inferencias, es el que le indica al agente que explore estas posibilidades y es el que se desarrolla a continuación.

6.3.1 Segmentación de la Secuencia Original

Como se mencionó anteriormente, representar la secuencia acción-cuadro por acción-cuadro no es ilustrativo. Para obtener un código de CLIPS más

compacto, se desea que el sistema no sólo detecte que puede ejecutar ciertas acciones en órdenes distintos, si no que también agrupe aquellas que forzosamente deben ser ejecutadas en forma secuencial. Para ello se utiliza el árbol descrito en la figura 6.1.

Existen dos formas de realizar este análisis:

Una puede considerarse como "en tiempo real". El agente requiere presenciar la misma acción pero siendo ejecutada en órdenes distintos. A partir de la representación que ya guardó de la secuencia, puede ir comparando si puede predecir el paso siguiente y registrar una división cada vez que el orden sea alterado.

La otra opción es que el agente genere el árbol con la información que ya obtuvo al presenciar la primera secuencia. Determinar el orden estrictamente requerido, puede verse como un problema de composición de funciones. Para componer dos funciones se requiere que el dominio de una sea la imagen de la que ha de aplicarse primero. Del mismo modo, para colocar un zapato se requiere tener primero al pie con el calcetín puesto. Las demás alteraciones al orden general son irrelevantes. Todo esto lo puede aplicar el agente, ya que cuenta con las descripciones de las entradas y las salidas de cada acción-cuadro.

En un agente más avanzado, sería deseable incluir ambos métodos. Inclusive se puede proponer que el agente detecte secuencias con mayores diferencias (no sólo alteraciones en el orden), pero que lleven a los mismos fines. Podría incluso hacer comparaciones de manera que grabe, como su secuencia privilegiada, aquella más corta y sólo guardar versiones resumidas de otras que pudiera requerir según su contexto. Sin embargo, se dejan todas estas posibilidades como desarrollo a futuro.

Para esta implementación, se optó por el segundo método.

Dado que se cuenta con la caracterización de los cambios entre cuadro y cuadro, considerando exclusivamente aquellos objetos que tomaron parte, es posible que el agente tome todos los objetos requeridos en el mundo y, al estilo de lo que hará CLIPS cuando ejecute las reglas finales, vaya determinando qué acciones se pueden realizar dado un estado del mundo.

El algoritmo para generar el árbol es el siguiente:

Entradas: Lista de tipos de los objetos presentes al inicio.

Secuencia inicial de acciones que, se sabe, alcanza el objetivo.

Meta (listado de los objetos que deben estar presentes en el mundo, a la salida de la ejecución)

- Si los objetos listados en meta ya se encuentran en el mundo, terminar.

- Para cada acción reportada en la secuencia inicial, verificar si es realizable sobre la configuración inicial del mundo.
  - Se verifica si es realizable, tratando de eliminar de la representación del mundo, tantos objetos como sean requeridos por la acción.
    - Si se encuentran todos los requeridos, devolver cómo quedaría el mundo, después de ejecutar la acción.
    - Si no, reportar que no es realizable.
  - Cuando la acción sea realizable, llamar de nuevo a este método con los argumentos siguientes: el mundo modificado, la lista de acciones por realizar menos la acción elegida. Al resultado devuelto, se le antepone la acción elegida.
- Devolver la estructura con todas las rutas probable.

La forma concreta de almacenar la información es cuestión de la implementación. Para este experimento se utilizaron nodos de un árbol. Dentro del nodo se almacena la acción realizada y los hijos apuntan a todas las rutas que se pueden seguir después de la acción en el nodo.

El resultado que se obtiene es precisamente el árbol de la figura 6.1.

Una vez que se ha generado este árbol, el sistema lo recorre en profundidad, generando listas con subsecuencias de acciones continuas. Cada vez que detecta algún nodo con más de un hijo, genera un segmento distinto para cada rama, correspondiendo a una subsecuencia de acciones-cuadro. Al regresar a analizar otras rutas posibles, cada vez que llega a una ruta que sigue un cause distinto a lo predicho por el segmento generado previamente, lo vuelve a segmentar. Este procedimiento se repite hasta dejar una serie de listas con las acciones que deben ser ejecutadas en estricto orden.

Este algoritmo queda mejor expresado en las figuras siguientes, que detallan los pasos relevantes al aplicar el procedimiento sobre el árbol de la figura 6.2:

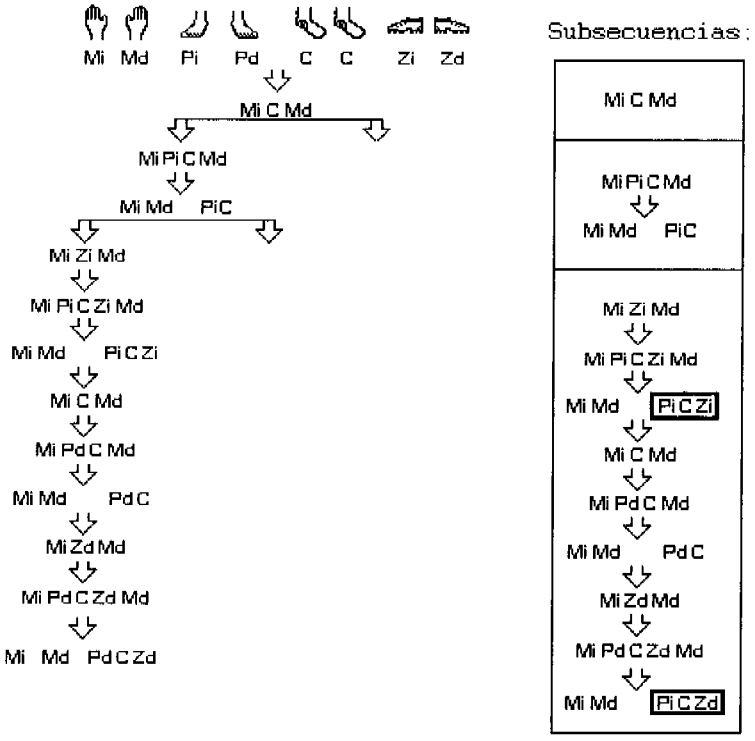


Figura 6.2 Al recorrer la primera rama, el agente detecta dos puntos de bifurcación y por lo tanto genera tres subsecuencias.

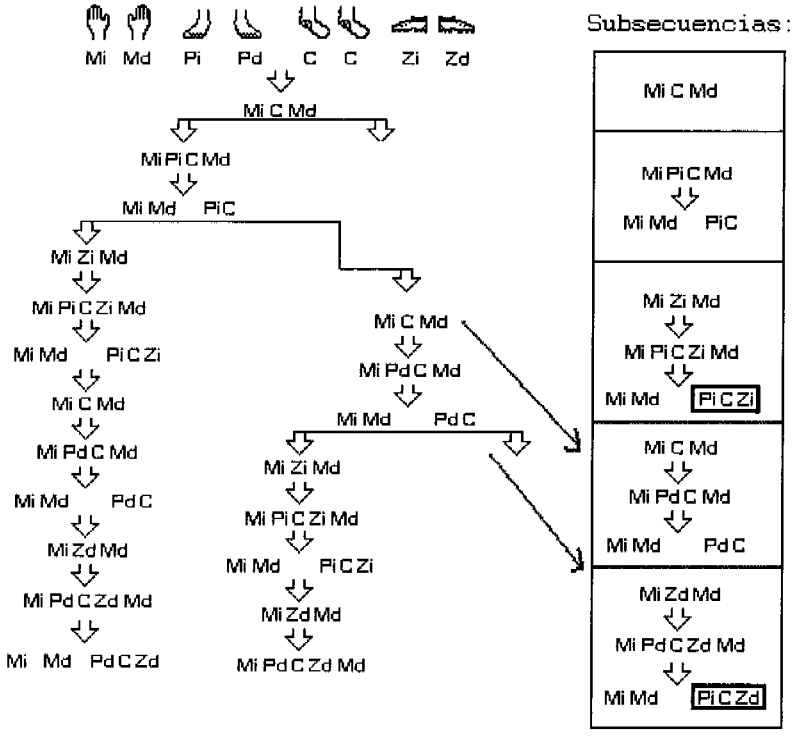


Figura 6.3 Al recorrer la segunda rama, detecta aquellas acciones que deberían encontrarse a la cabeza de una subsecuencia y no en medio (es decir, vuelve a segmentar). Además considera las nuevas bifurcaciones del árbol.

Aunque ya para la segunda rama se detectaron todos los segmentos que podrían encontrarse, el algoritmo no puede saberlo de antemano, así que continúa ejecutándose hasta terminar de recorrer el árbol.

Cada uno de estos segmentos puede ser traducido directamente a reglas de CLIPS. Se eliminan los segmentos idénticos ya que CLIPS no requiere que se le indique cuántas veces ejecutar cada uno, la máquina de inferencias lo determina.

6.4 Extracción de Reglas

El proceso de análisis descrito en las secciones anteriores permite extraer las reglas para el entorno CLIPS en una forma muy natural. Concretamente, ya sólo se requiere acomodar la información según la sintaxis de CLIPS. Esta sintaxis se explica en el Apéndice A.

Las plantillas requeridas se definen a partir de las clases participantes. Por motivos de expresividad, se eligió que, si las clases tienen padre, el nombre de la relación esté dado por la clase padre; mientras que la especificación de qué hijo se trata, es una rendija. Además, si participaran varios objetos indistinguibles, simultáneamente, se requiere especificar de qué instancia se trata. Esta distinción se hace agregando una rendija "instancia" cuyo valor es un identificador único para cada instancia (en este caso, un entero).

El estado inicial del mundo incluye todos los objetos identificados como argumentos de la secuencia, más los auxiliares.

Para cada segmento identificado se genera una regla. Las entradas de los segmentos se acomodan en los patrones de entrada de una regla de CLIPS (lado izquierdo de la regla). Del lado derecho, se pide que se eliminen los hechos del mundo, correspondientes a los argumentos, y se agregan hechos que indiquen los estados nuevos.

El resultado obtenido en este experimento es el siguiente:

```
;; Guio'n generado automa'ticamente para la ma'quina de inferencias CLIPS.
;; OBJETIVO: Ponerse los zapatos


; Objetos participantes:

(deftemplate ZCP (slot hijo))
(deftemplate ZAPATO (slot hijo))
(deftemplate CP (slot hijo))
(deftemplate PIE (slot hijo))
(deftemplate C#248 (slot hijo))
(deftemplate MANO (slot hijo))
(deftemplate CAL (slot instancia))

; Listado de los objetos presentes al inicio de la accio'n:

(deffacts estado-inicial
```

```
    (CAL (instancia 1))
    (CAL (instancia 2))
    (PIE (hijo PD))
    (PIE (hijo PI))
    (ZAPATO (hijo ZD))
    (ZAPATO (hijo ZI))
    (MANO (hijo MD))
    (MANO (hijo MI))
    (META Ponerse los zapatos))

; Segmentos identificados

(defrule accion1
    (META Ponerse los zapatos)
    ?f1<- (CAL (instancia ?instancia))
    ?f2<- (MANO (hijo MD))
    ?f3<- (MANO (hijo MI))
    =>
    (retract ?f3 ?f2 ?f1)
    (assert (C#248 (hijo C#246)))
)

(defrule accion2
    (META Ponerse los zapatos)
    ?f1<- (C#248 (hijo C#246))
    ?f2<- (PIE (hijo PD))
    =>
    (retract ?f2 ?f1)
    (assert (CP (hijo CPD)))
    (assert (MANO (hijo MD)))
    (assert (MANO (hijo MI)))
)

(defrule accion3
    (META Ponerse los zapatos)
    ?f1<- (CP (hijo CPD))
    ?f2<- (ZAPATO (hijo ZD))
    =>
    (retract ?f2 ?f1)
    (assert (ZCP (hijo ZCPD)))
)

(defrule accion4
    (META Ponerse los zapatos)
    ?f1<- (C#248 (hijo C#246))
    ?f2<- (PIE (hijo PI))
    =>
    (retract ?f2 ?f1)
    (assert (CP (hijo CPI)))
    (assert (MANO (hijo MD)))
    (assert (MANO (hijo MI)))
)

(defrule accion5
    (META Ponerse los zapatos)
    ?f1<- (CP (hijo CPI))
    ?f2<- (ZAPATO (hijo ZI))
    =>
    (retract ?f2 ?f1)
    (assert (ZCP (hijo ZCPI)))
)


; Objetivo alcanzado

(defrule termina
  ?f0 <- (META Ponerse los zapatos)
    (ZCP (hijo ZCPI))
    (ZCP (hijo ZCPD))
    =>
    (retract ?f0))
```

El código generado lista a todos los objetos participantes e incluso a un grupo intermedio, describe el estado inicial y luego describe cuatro acciones, más una regla que indica, explícitamente, que el objetivo ha sido alcanzado. Las cuatro acciones generadas corresponden a:

- Acción 1: Tomar un calcetín con ambas manos. El tipo asignado a esta configuración fue C#246[4], hija de C#248 (que contiene a ambas manos).
- Acción 2: Meter el calcetín en el pie derecho.
- Acción 3: Calzar el zapato derecho.
- Acción 4: Meter el calcetín en el pie izquierdo.
- Acción 5: Calzar el zapato izquierdo.

Una vez reunida esta información, el programa se puede ejecutar en CLIPS. Dependiendo del criterio de selección de reglas, la máquina de inferencias aplicará las reglas en órdenes distintos. De este modo, CLIPS puede generar secuencias en las que se pone primero ambos calcetines y luego los zapatos o que inicie primero vistiendo completamente un pie y luego el otro.

Usualmente, los sistemas expertos reciben las reglas de un ser humano que las extrae y redacta primero. Resulta, entonces, de interés, comparar el código obtenido, con el código generado por una persona. Es importante mencionar que: el código que se muestra a continuación, fue generado antes de desarrollar el sistema artificial y fue utilizado como guía para diseñar los diversos módulos y algoritmos que lo conforman. Por esta razón, varias de las similitudes entre ambos códigos son, más bien, una consecuencia lógica del origen de ambos. Sin embargo, también emergen algunas diferencias que es importante analizar. (La comparación en detalle de ambos códigos se llevará a cabo en el capítulo siguiente, por el momento sólo se analiza cada uno por separado.)

[4] Los identificadores asignados a las clases generadas automáticamente varían en cada ejecución del programa, por lo que estos números no coinciden con los de la ejecución expuesta en el capítulo 4 y detallada en el apéndice B.

```
(deftemplate pie (slot lado))          ; Objetos participantes
(deftemplate zapato (slot lado))
(deftemplate calcetín (slot instancia))
(deftemplate pie_calcetín (slot lado))
(deftemplate pie_calcetín_zapato (slot lado))

(deffacts estado-inicial               ; Descripción del estado inicial
 (pie (lado derecho))
 (pie (lado izquierdo))
 (zapato (lado derecho))
 (zapato (lado izquierdo))
 (calcetín (instancia 1))
 (calcetín (instancia 2))
 (goal ponerse-zapatos)                ; Intención
)

(defrule poner_calcetín                ; Ponerse el calcetín en un pie.
 (goal ponerse-zapatos)
 ?f1 <- (pie (lado ?lado))             ; El pie debe estar desnudo.
 ?f2 <- (calcetín (instancia ?instancia)); Tomar un calcetín
 =>
 (retract ?f1 ?f2)                     ; El pie ya no está desnudo y el
                                       ; calcetín ya fue utilizado.
 (assert (pie_calcetín (lado ?lado)))); Pie con calcetín.

(defrule poner_zapato                  ; Zapato sobre pie con calcetín.
 (goal ponerse-zapatos)
 ?f1 <- (pie_calcetín (lado ?lado))
 ?f2 <- (zapato (lado ?lado))
 =>
 (retract ?f1 ?f2)
 (assert (pie_calcetín_zapato (lado ?lado))))

(defrule termina                       ; Objetivo alcanzado
 ?f1 (goal ponerse-zapatos)
 ?f2 (pie_calcetín_zapato (lado derecho))
 ?f3 (pie_calcetín_zapato (lado izquierdo))
 =>
 (retract ?f1))
```

Analizando en detalle los elementos del programa anterior, es posible inferir cómo es que un ser humano pudo haber llegado a plantear la solución en estos términos.

El primer elemento del programa de CLIPS define los objetos que participarán a lo largo de toda la acción. Aquí se plantea la posibilidad de que el pie, el pie con calcetín y el pie con calcetín y zapato sean considerados objetos distintos. Este criterio permitirá indicar claramente, en la definición de las reglas, el orden parcial requerido para su aplicación.

El segundo elemento es la descripción del estado inicial: indica qué objetos se encuentran disponibles, cuyos estados serán modificados en las acciones subsiguientes, además se agrega la intención de colocar los zapatos. (Tal vez alguna persona descansando en sábado, en un día de calor, prefiera andar descalza, aunque tenga todos los elementos a la mano; se eliminan por el momento aquellos individuos que gustan de usar los zapatos sin calcetines). Esta misma descripción funciona como activador para la primera acción: poner el calcetín, lo cual será un prerrequisito para calzar los zapatos.

La regla que se refiere a poner el calcetín es activada cuando se cumplen tres hechos en el mundo: hay un pie (desnudo), un calcetín y el deseo de ponerse los zapatos. Puede decirse que esta regla activa el instinto reflejo de poner el calcetín en el pie. Esta regla, dados los hechos iniciales, será colocada en la agenda dos veces: una para el pie derecho y alguna instancia de calcetín y otra para el pie izquierdo y el calcetín restante. (Obsérvese que se está considerando a ambos calcetines como objetos indistinguibles, por lo que sólo se les asocia un identificador que permita darle seguimiento a cada instancia, pero se les trata indistintamente).

Al ejecutarse dicha regla, los hechos cambian para manifestar que ya no hay pies desnudos o calcetines, sino pies con calcetines. Al agregar este nuevo hecho a la descripción del mundo se activa el siguiente "instinto": poner el zapato. Obsérvese que ésta no podría haber sido activada antes, pues requiere de pies **con** calcetines. Considerar al pie con el calcetín como un objeto distinto a sus dos componentes permite conseguir el efecto deseado en este lenguaje: que los zapatos queden **sobre** los calcetines (calcetines puestos). La máquina nunca intentará hacerlo de otro modo.

Otro aspecto importante radica en el uso de las rendijas "lado". Las acciones requeridas para vestir un pie son exactamente las mismas que para vestir al otro, por lo que no es necesario redefinirlas para cada uno, sólo es necesario llevar el seguimiento de sobre quién ya fueron aplicadas y de ello se encarga CLIPS.

Por último, es importante recalcar la manifestación de lo obvio: las acciones poner calcetín y poner zapato (aunque relacionadas por ser el argumento de una, la respuesta de la otra), son acciones independientes; por lo que no es necesario ejecutarlas una estrictamente después de la otra, es posible intercalar otras acciones entre ellas y esto es lo que da origen a tantos órdenes posibles para alcanzar el objetivo final de vestir ambos pies.

La regla final sólo fue puesta para indicar que el deseo inicial ha sido satisfecho y por lo tanto el objetivo puede ser retirado de la lista de hechos. Ésta es una manera de incluir en CLIPS el concepto de "satisfacción" en un estilo más intuitivo, ya que el sistema computacional, más bien, queda "satisfecho" cuando no hay más reglas pendientes en la agenda; pero esto no necesariamente corresponde con la satisfacción del objetivo introducido por el programador. En realidad, la meta "poner zapatos" fue introducida artificialmente como un hecho más del mundo y sólo se puede comprobar que ha sido alcanzada verificando el estado al que ha llegado el sistema tras la aplicación de las reglas en la agenda. Este detalle ha sido incluido con la finalidad de volver más elegante el programa, para ser más consistente con el análisis del pensamiento humano.

CAPÍTULO VII

ANÁLISIS DE RESULTADOS Y CONCLUSIONES

7.1 ANÁLISIS

7.1.1 La Jerarquía de Clases

Se enfrentó el problema de generar una jerarquía de tipos. El resultado obtenido fue satisfactorio para el problema central.

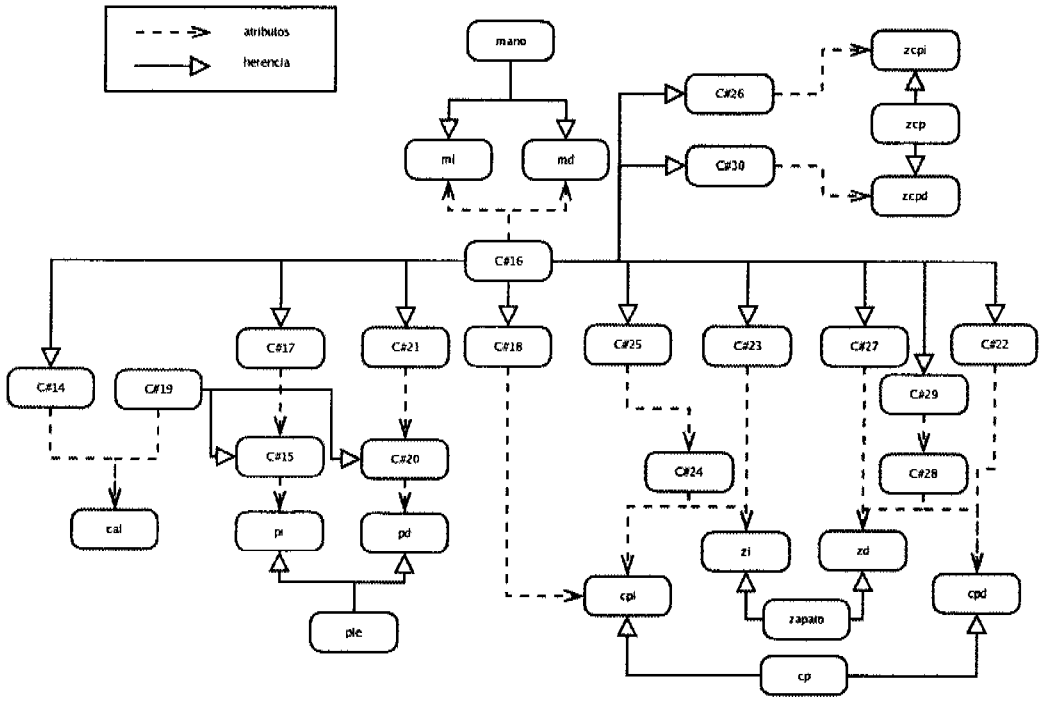


Figura 7.1 (Repetición de la Figura 4.12) La jerarquía de clases generada automáticamente para los objetos y grupos participantes en una secuencia que describe cómo son puestos los calcetines y los zapatos.

Como se pudo ver a lo largo de este trabajo, la categorización de los objetos y grupos de ellos, que observó el agente, ayudó al posterior análisis de las interacciones entre ellos. La asociación de un tipo que pudiera ser leído, sin la necesidad de entrar en detalles, agilizó la aplicación de los diversos algoritmos.

Sin embargo, el sistema utilizado carece de expresividad. La primera consecuencia de esta limitante fue que, para reconocer el tipo de la imagen, fue necesario descender hasta los átomos, para luego llegar a lo más general. En un sistema real, la finalidad de la jerarquía es, precisamente, evitar entrar en detalles. Se ha considerado la posibilidad de indexar cada

clase por un conjunto de atributos característicos, que permanezcan inalterados a pesar de pequeñas transformaciones en el estado del objeto. Una vez identificada la clase más general (la que requiere un conjunto menor de atributos para ser identificada), se va requiriendo más información para afinar la búsqueda.

Si bien, por el momento se sobresimplificó el sistema de clases intencionalmente, es indispensable retirar esta limitante para continuar avanzando en esta investigación.

Relacionado a la falta de expresividad, se encuentra la reducida representación espacial. Para este trabajo fue posible incluir la posición del objeto (pertenencia o no pertenencia al grupo) dentro del tipo de la clase. En sentido estricto, esto no es correcto; sin embargo sí fue muy útil para caracterizar situaciones y darle seguimiento a las interacciones de los objetos en el mundo.

A este respecto, un sistema más maduro tendrá que caracterizar las situaciones y los objetos que participan en ellas, dentro de categorías diferentes. Es probable que se deba definir un tipo distinto de relación para expresar estos hechos. Sin embargo, queda claro que aún esta nueva relación debe poder beneficiarse de los mismos criterios de herencia, pues fueron los que condujeron el análisis de los movimientos. Al separar las categorías e incrementar la información espacial a 3 dimensiones, el seguimiento de los movimientos de los objetos indistinguibles se vuelve mucho más sencillo, pues puede tomarse en cuenta su posición con mayor facilidad.

Por último, resta mencionar la limitante más grande: el algoritmo implementado no es capaz de reestructurar la jerarquía cuando hay varios ancestros involucrados. Al incluir el uso de sobrecarga, se abrirían nuevas posibilidades para atacar este problema.

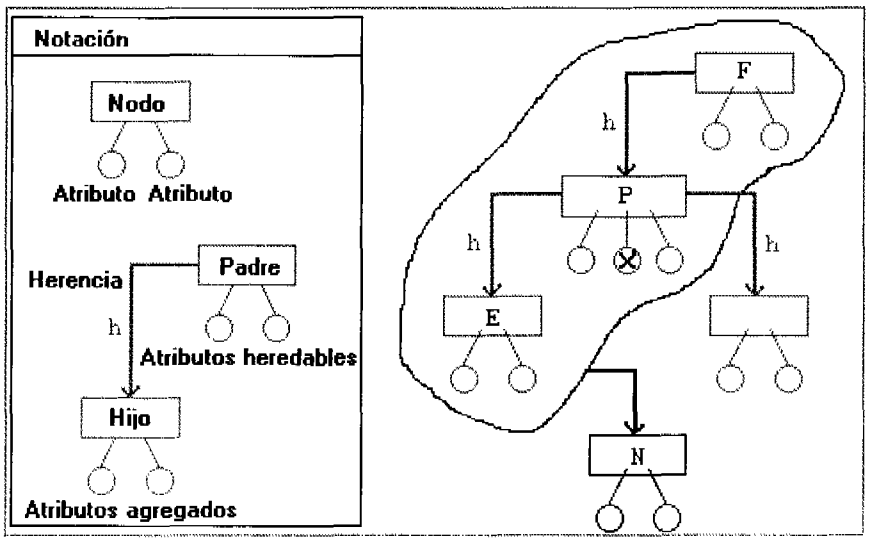


Figura 7.2 (Repetición de la figura 4.10) El nuevo grupo comparte casi todos sus atributos con el grupo E, excepto uno que pertenece a un ancestro y dos que debe agregar a su descripción.

En definitiva, los puntos aquí señalados abren toda una rama de investigación. La aproximación de este trabajo ha consistido, únicamente, en iniciar la exploración de su tratamiento. Ha puesto de manifiesto la importancia de continuar con su desarrollo, para alcanzar un manejo de más alto nivel del conocimiento, y ésta es su principal contribución.

7.1.2 Seguimiento de Instancias

Se expuso la metodología utilizada para seguir la actuación de los objetos del mundo, en términos de instancias internas. Se plantearon dos alternativas para el análisis y aprendizaje de secuencias: análisis a partir de una sola secuencia y actualización frente a experiencias nuevas (ver 6.3.1). Solamente se implementó una, por lo que se propone extender el sistema con la propuesta faltante.

Se obtuvieron funciones que actualizan la representación interna del mundo, de un cuadro a otro, al identificar los tipos de los objetos participantes al inicio y al final. El siguiente paso es aplicar el mismo concepto pero a acciones que involucren varios cuadros: identificar cuadros que correspondan a estados iniciales de acciones previamente aprendidas, seguir el desarrollo cuadro a cuadro para confirmar, recuperarse si se perdió la visión de algún cuadro al reconocer los pasos siguientes, así sea el estado final. Generar funciones que describan acciones cada vez más complejas, en términos de las más sencillas.

7.1.3 Instancias Permanentes

Una extensión más, se plantea en el manejo de las instancias. Por el momento, las instancias fueron generadas y eliminadas para cada ocasión en que se presencia la secuencia. Se desea poder crear instancias más permanentes: objetos que puedan reaparecer en varios guiones y ser identificados, al igual que las clases, por aquellos atributos permanentes que los distingan de los demás.

Incluir estas consideraciones añadiría la posibilidad de generar todo un contexto, el historial de cada objeto que conoce el agente y darle seguimiento. Por supuesto, el sistema tendría que ser flexible: poder reestructurar su información si descubre que su asignación de instancias fue errónea.

Por ejemplo, una instancia permanente podría corresponder con una persona conocida por agente. Cada vez que se encontrara con la misma persona, debería ser capaz de identificarla y ligar sus nuevas experiencias a la misma instancia.

Si dicha persona decidiera teñirse el cabello, su descripción debería ser modificada, pero se le seguiría asociando la misma instancia.

7.1.4 Lo que Indican los Guiones

Si comparamos los códigos de CLIPS generados, por el desarrollador del sistema y por el agente (capítulos 6), se observan diferencias interesantes.

```
Humano:
(defrule poner_calcetín                    ; Ponerse el calcetín en un pie.
 (goal ponerse-zapatos)
 ?f1 <- (pie (lado ?lado))                 ; El pie debe estar desnudo.
 ?f2 <- (calcetín (instancia ?instancia)); Tomar un calcetín
 =>
 (retract ?f1 ?f2)                         ; El pie ya no está desnudo y el
                                           ; calcetín ya fue utilizado.
 (assert (pie_calcetín (lado ?lado))));Pie con calcetín.
```

```
Agente:
(defrule accion1
   (META Ponerse los zapatos)
   ?f1<- (CAL (instancia ?instancia))
   ?f2<- (MANO (hijo MD))
   ?f3<- (MANO (hijo MI))
   =>
   (retract ?f3 ?f2 ?f1)
   (assert (C#248 (hijo C#246)))
)

(defrule accion2
   (META Ponerse los zapatos)
   ?f1<- (C#248 (hijo C#246))
   ?f2<- (PIE (hijo PD))
   =>
   (retract ?f2 ?f1)
   (assert (CP (hijo CPD)))
   (assert (MANO (hijo MD)))
   (assert (MANO (hijo MI)))
)
```

Tabla 7.1 Segmentos de los códigos generados por un humano y por el agente, donde se ilustran las diferencias más notorias.

En primer lugar, se encuentra la separación de la acción de poner el calcetín. El código del desarrollador considera la acción completa "poner el calcetín" incluyendo el agarrarlo. Mientras, la máquina, separa "agarrar el calcetín" de "poner el calcetín". Esto ocurrió porque la máquina considera el

hecho de que, tras haber tomado el calcetín, aún se puede decidir en qué pie ponerlo (al menos la primera vez).

Otra diferencia muy importante está en que, en el código del desarrollador, las acciones como "poner zapato derecho" y "poner zapato izquierdo" son agrupadas en una sola. Y esta diferencia es la que más nos dice acerca de las extensiones que conviene hacer al procedimiento planteado.

En primer lugar, se detecta que el código generado indica que se encuentran los dos tipos de manos, pies y zapatos, y las acciones que se ejecutan sobre ellos son similares. Sin embargo, el agente aún no es capaz de agruparlas. La principal causa de ello se encuentra, una vez más, en la poca expresividad del sistema de tipos propuesto.

Como se señaló al principio, el sistema de tipos no permite indicar mayor información acerca de las características de los objetos observados. En particular, el agente no sabe que existe algo en común entre "pi" y "zi" y ese algo es que se encuentran del lado izquierdo. De la secuencia que poseé el agente, es posible, con un poco más de análisis, detectar que poner el calcetín en el pie izquierdo está directamente ligado con poner el calcetín en el zapato izquierdo. Sin embargo, el agente no tiene manera de saber qué es lo que tienen estos dos objetos en común, que obliga a seguir este orden y que es lo mismo que obliga a poner el zapato derecho sobre el pie derecho. Las reglas indican que es así (de otro modo marcan que no se han cumplido las precondiciones), pero no se sabe porqué.

En principio, el agente podría quedarse al nivel de detectar que existe "algo" que los une, pero no valía la penar agregar los métodos para hacerlo por dos razones:

- Aún cuando lo detectara, no podría agregar ese descubrimiento a la información en la jerarquía de clases. Para realizar esta extensión de manera que valga la pena, es prerrequisito indispensable extender el manejo de esta jerarquía. Se requiere incluir, ahora sí, todo el poder expresivo de la teoría de marcos; utilizando diferentes tipos de atributos, sobrecarga, etc. Además, se requiere ampliar el criterio utilizado para generar la jerarquía automáticamente. Claramente, este es un problema completamente abierto. Pero un objetivo de esta tesis ha sido, precisamente, comenzar a esbozar cómo realizar esta tarea y cómo puede utilizarse en las fases siguientes del seguimiento de acciones y su aprendizaje.

- Esta generalización requiere de buscar "qué tienen en común todos los objetos participantes". Se trata de encontrar la relación existente entre todos ellos, para, ahora sí, agregar la rendija correspondiente. Agregar el código a este ejemplo, sería agregar rutinas demasiado sesgadas, que, en este punto, ya no contribuyen con información relevante.

El siguiente paso, ya con un manejo de tipos más expresivo, consistiría en asignar tipos a las secuencias de acciones. Estos tipos deberían permitir encontrar las similitudes entre diferentes secuencias, tal y como se hizo para

los grupos de objetos. Entonces, secuencias similares ejecutadas sobre objetos de tipos hermanos, pueden ser unificadas, creando funciones que admitan a cualquiera de los hermanos y sepan cómo comportarse con cualquiera de ellos. Esto, en lugar de listarlas por separado, como se ha logrado hasta este momento.

Por último, no puede dejarse de comparar el código obtenido con el código de alguien ajeno al diseño del sistema. Se incluye otro código en la tabla 7.2.

```
;*************************************************************************
;*   zapatos.clp
;*   FI-UNAM    J. Savage    4 09-2003
; *************************************************************************
;*   Hechos iniciales
(deffacts estado-inicial
    (estado levantándose)
    (pantalón puesto)
    (pie derecho desnudo)
    (pie izquierdo desnudo)
    (calcetín derecho)
    (calcetín izquierdo)
    (zapato derecho)
    (zapato izquierdo)
)
;*   REGLAS
; Regla inicial
(defrule inicio
    (estado levantandose)
    (pantalón puesto)
    (pie ?pie desnudo)
    =>
    (assert (colocar zapato ?pie)
)
; Coloca el calcetín
(defrule colocar-calcetín
    (colocar zapato ?pie)
    ?f1 <- (pie ?pie desnudo)
    ?f2 <- (calcetín ?pie)
    =>
    (retract ?f1 ?f2)
    (assert (poner calcetín ?pie))
)
; Coloca el zapato
(defrule colocar-zapato
   ?f1<- (colocar zapato ?pie)
   (not (pie ?pie desnudo))
```

```
        ?f2<- (zapato ?pie)
        =>
        (retract ?f1 ?f2)
        (assert (poner zapato ?pie))
    )
```

Tabla 7.2 Código propuesto por alguien independientemente del desarrollo del sistema.

Los códigos expuestos inicialmente y éste último tienen grandes diferencias, incluso en el planteamiento del problema. Las reglas mostradas en la tabla 7.2 fueron planteadas mucho antes de iniciar el trabajo de esta tesis. Claramente, no había intensión de que un sistema pudiera extraerlas.

El código de la otra persona recurre a nociones enteramente humanas como: desnudo, levantarse y los calificativos "derecho" e "izquierdo". El sistema no conoce nada de esto, por lo que nunca expresará las reglas en estos términos. El sistema se "expresa" en términos de las cosas que conoce: las características de las imágenes. Los seres humanos podemos interpretar su lenguaje, pero no a la inversa.

Además el código humano considera más aspectos relativos a la acción de vestirse, como el hecho de que es necesario tener un pantalón puesto. (Claro que, si fuese una mujer o un escocés con falda, la regla ya no aplica). El sistema no menciona nada al respecto porque ni siquiera lo ha "visto". Sin embargo, sería posible introducir una secuencia de imágenes más larga, que describiera el proceso completo de vestirse. En este caso el sistema también podría incluir reglas que mencionen condiciones sobre otras prendas.

Otra diferencia importante, pero cuya manifestación es más sutil, es que el código humano es muy particular. El lenguaje, las reglas y el planteamiento en general lucen ad hoc al problema de ponerse los calcetines y los zapatos, pero no se ve una estructura regular fácilmente generalizable a otro tipo de reglas. Los otros códigos, son más uniformes.

Estas diferencias se derivan de las capacidades de los seres humanos para manejar grandes cantidades de conocimiento, de todo tipo, y dar un tratamiento particular a cada cosa, a la vez que da por sentado su contexto.

A todas estas diferencias se suman las que ya se habían citado en el análisis anterior.

7.2 CONCLUSIONES

7.2.1 Análisis de Objetivos

Este fue el objetivo planteado al inicio de esta tesis:

El objetivo primordial de esta tesis fue estudiar la extracción de reglas para un planeador a partir de ejemplos concretos que recibe un agente.

Se cubrió el objetivo de esbozar el procedimiento para hacer que un agente analice y aprenda el procedimiento para realizar acciones nuevas, considerando diversos órdenes de ejecución.

Se bosquejó el diseño de un sistema de visión a partir de investigaciones exitosas actuales, para que funja como módulo de entrada para el sistema. Esta propuesta puede utilizarse como guía para coordinar los esfuerzos realizados en la construcción de robots móviles que aprenden y comprenden su entorno.

Dado este módulo de visión, el seguimiento de las acciones se realizó sin poseer información semántica, acerca de las imágenes percibidas y las interacciones entre ellas. Con esto, se presentan las bases para un agente que pueda aprender a partir de la información que recibe a través de sus sensores, sin que los conceptos deban ser introducidos manualmente. Por ejemplo, un concepto como el de "zapato" se irá generando a partir del análisis de la interacción de la silueta reconocida, con las demás imágenes que se reconocen en su entorno. Ciertamente, el diseño es aún muy elemental, pero puede ser extendido para cubrir aspectos adicionales.

Se logró realizar, aunque fuera sólo por la duración de cada secuencia, el seguimiento de "un mismo objeto". En un sistema más avanzado, se podrían combinar técnicas de análisis de imágenes basadas, por ejemplo, en el uso de cadenas de Markov, que no "razonan" (es decir, no manejan conceptos), con una versión adaptada de este algoritmo, que utiliza conocimiento con un mayor nivel de abstracción, para resolver ambigüedades.

Tras el análisis detallado de las secuencias de imágenes simuladas, presentadas al agente, se obtuvieron reglas para el lenguaje CLIPS. Tradicionalmente, reglas de alto nivel de abstracción, como lo son las reglas utilizadas por una máquina de inferencias, son generadas por seres humanos, pues requieren analizar el contexto del cual se extraen. Este sistema incursiona directamente en este campo, logrando generar sus propias reglas a partir de un experimento sencillo.

Ampliar este sistema para que pueda ser utilizado para atacar problemas de la vida real es posible. Sin embargo, será necesario tomar conciencia de las limitaciones debidas a la complejidad en tiempo y espacio de estos problemas, utilizar heurísticas a partir de puntos que aún están por definir. Será importante continuar el diseño, utilizando los desarrollos que consumen más recursos, sólo para analizar parte de los problemas generales, el resto tendrá que renunciar a la obtención de soluciones óptimas.

Precisamente, previendo el incremento en la complejidad, es que se ha propuesto el uso de la jerarquía de clases, para el reconocimiento y manejo de objetos; así como la agrupación de acciones en secuencias, subrutinas que pueden utilizarse, a su vez, como elementos de otras secuencias. Estos agrupamientos pueden darse en todos los niveles; el algoritmo diseñado no impone un límite predefinido.

7.2.2 Contribución

En esta tesis se puso gran énfasis en diseñar un agente virtual cuyo funcionamiento pueda ser extendido a robots que interactúen con seres humanos. Se observó que la reducción de información compleja a manipulaciones simbólicas, viabilizan la generación de reglas que de otro modo no se podrían extraer.

También se observó cómo es posible generar dinámicamente "conceptos" nuevos a partir de agrupaciones típicas de objetos reconocibles y extraer reglas de comportamiento entre ellos. Estas características forman la base de una definición de las cosas extraída de la experiencia. De continuarse con la extensión de este trabajo, podría definirse la arquitectura de un agente capaz de trabajar con contextos, pues sus definiciones vendrían de su experiencia de las cosas. Un agente de este tipo podría enfrentarse con mayor naturalidad a los problemas que plantea la interacción amigable humano–máquina.

7.2.3 Trabajo Futuro

Se proponen como puntos prioritarios para continuar la investigación:

- Extender el manejo y generación automática de la jerarquía de clases, a partir de la experiencia. Se recomienda introducir la posibilidad de que el agente reciba retroalimentación para evaluar qué tan buenas fueron sus decisiones al diseñar las categorías.

- Ahondar en la detección de patrones usando **sistemas simbólicos**. Es decir, encontrar relaciones, aún no expresadas, entre diversos objetos en la jerarquía de clases. Estas relaciones deberían buscarse tanto en la jerarquía, como en los historiales de las instancias (situaciones en las que los objetos se han visto involucrados).

Todo esto permitiría introducir un amplio manejo de contextos por el agente.

APÉNDICE A

CLIPS

CLIPS es un lenguaje de programación diseñado en el Centro Espacial Johnson de la NASA. Su nombre es un acrónimo de "Sistema de Producción Integrado al Lenguaje C" (debido a la traducción del inglés las siglas han quedado al revés). La versión 5.0 de este lenguaje provee soporte para programación basada en reglas, orientada a objetos y procedimental.

En lo que respecta al soporte para un sistema experto basado en reglas, CLIPS está compuesto por:

- Una lista de hechos. Es la memoria global que almacena la información del sistema. Ésta es la descripción del estado actual del mundo.
- Una base de conocimientos. Contiene todas las reglas, que son el medio para definir las acciones en este lenguaje.
- Motor de inferencias. Controla la ejecución del sistema haciendo uso de encadenamiento hacia adelante.

Los hechos concretos son representan mediante el **nombre de la relación** seguido de cero o más **campos con nombre** o **rendijas** y sus valores asociados. Por ejemplo:

```
(persona (nombre "Ángel Miramar") (edad 29) (color-de-ojos verde)
(color-del-cabello negro))
```

El orden en el que se especifican las rendijas no es importante.

Antes de que se puedan introducir hechos, CLIPS requiere que se le indique la lista de rendijas válidas para un nombre de relación dado. Grupos de hechos que comparten el mismo nombre de relación y contienen información común pueden describirse mediante el constructor *deftemplate* (define una *plantilla*, para la relación). El formato de este comando es:

```
(deftemplate <nombre-de-la-relación> [<comentario-opcional>]
(slot <nombre-de-la-rendija>)*)
```

Utilizando esta sintaxis, la estructura del hecho *persona* puede describirse de la manera siguiente.

```
(deftemplate persona "Descripción de una persona"
  (slot nombre)
  (slot edad)
  (slot color-de-ojos)
  (slot color-del-cabello))
```

Es posible definir plantillas para hechos sin rendijas, éstos pueden servir como banderas y no es necesario usar el constructor *deftemplate*. (En

realidad este tipo de hechos pueden pertenecer a una categoría más de CLIPS, conocida como *hechos ordenados,* pero su uso está más allá del interés de este trabajo). Un ejemplo es el hecho:

```
(quiero-ponerme-los-zapatos)
```

Para agregar nuevos hechos a la descripción del estado actual del mundo se utiliza el comando:

```
(assert <hecho>+)
```

Para eliminarlos se utiliza:

```
(retract <índice-del-hecho>+)
```

Es posible introducir varios hechos a la vez mediante la construcción:

```
(deffacts <nombre-del-grupo-de-hechos> [comentario-opcional]
 <hechos>*)
```

Finalmente, las reglas que indican cuándo y cómo modificar el estado del mundo se definen mediante la siguiente construcción:

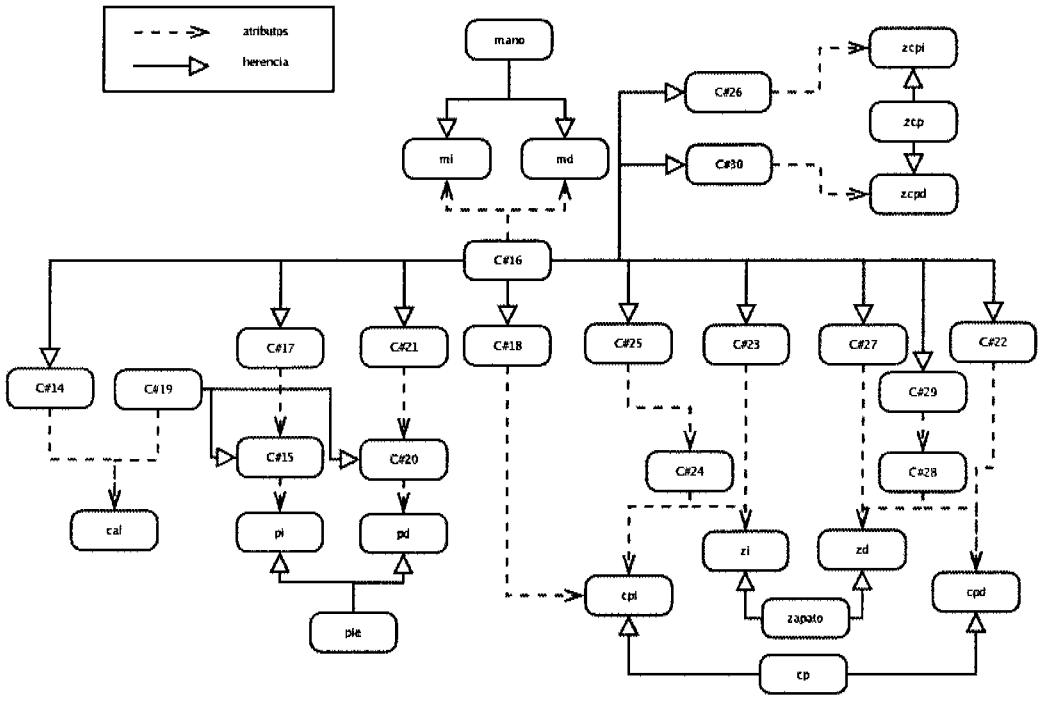
```
(defrule <nombre-de-la-regla> [comentario]
 <patrones>*    ; Lado izquierdo de la regla (LHS)
 =>
 <acciones>*)   ; Lado derecho de la regla (RHS)
```

El lado izquierdo de la regla indica las precondiciones, aquellos hechos que deben ser verdaderos en el mundo para que la regla sea disparada. En el lado derecho se especifican los predicados cuyo valor será modificado tras la aplicación de la regla.

Cada vez que todos los patrones indicados en una regla se cumplen en el mundo, la regla se activa y es puesta en una agenda. Al momento de ejecutar un programa, las reglas activas van siendo ejecutadas según su prioridad. En este trabajo, no se utilizó la asignación de prioridades explícitamente, por lo que no se abunda en este tema.

APÉNDICE B

INTERPRETACIÓN DETALLADA DE LA REPRESENTACIÓN INTERNA OBTENIDA EN EL EXPERIMENTO



Reproducción de la Gráfica 4.1 La jerarquía de clases generada automáticamente para los objetos y grupos participantes en una secuencia que describe cómo son puestos los calcetines y los zapatos.

Inicialmente, el sistema de visión simulado indica explícitamente la presencia de las clases:

- Mano, de la cual heredan mi y md.
- Cal.
- Pie, de la cual heredan pi y pd.
- Zapato, de la cual heredan zi y zd.
- Cp (calcetín con zapato), de la cual heredan cpi y cpd.
- Zcp (pie con calcetín y zapato), de la cual heredan zcpi y zcpd.

Las clases restantes son definidas por el sistema y caracterizan en qué forma interaccionan los objetos observados. Los nombres son generados automáticamente conforme van siendo creadas. A continuación se incluye una lista con sus significados.

Clase	Atributos	Ancestros
C#14	(mi cal md)	C#16
C#15	(cal pi)	C#19
C#16	(md mi)	
C#17	(md C#15 mi) = (md (cal pi) mi)	C#16
C#18	(mi cpi md)	C#16
C#19	(cal)	
C#20	(cal pd)	C#19
C#21	(mi C#20 md) = (md (cal pd) mi)	C#16
C#22	(md cpd mi)	C#16
C#23	(md zi mi)	C#16
C#24	(cpi zi)	
C#25	(md C#24 mi) = (md (cpi zi) mi)	C#16
C#26	(md zcpi mi)	C#16
C#27	(md zd mi)	C#16
C#28	(cpd zd)	
C#29	(md C#28 mi) = (md (cpd zd) mi)	C#16
C#30	(md zcpd mi)	C#16

Llama la atención la aparición reiterada de la clase C#16 como clase padre, lo cual resulta perfectamente comprensible al leer que se trata del grupo cuyos integrantes son la mano izquierda y la mano izquierda. De este modo, cada clase hija representa alguna imagen en la que se sostiene algo con ambas manos.

BIBLIOGRAFÍA

[1] http://www.everythingisnt.com/ "Bicycle riding robot unveiled" 2005.

[2] Terry Winograd, MIT AI Technical Report 235, February 1971 with the title Procedures as a Representation for Data in a Computer Program for Understanding Natural Language.

[3] SHRDLU en http://hci.stanford.edu/~winograd/shrdlu/

[4] Terry Winograd, "Procedures as a representation for data in a computer program for understanding natural language", MIT, 1971, pp 3.

[5] M.M. Mesulam, "From sensation to cognition", Brain, No. 121, pp. 1029-1030 (1998).

[6] M.M. Mesulam, "From sensation to cognition", Brain, No. 121, pp. 1021 (1998).

[7] D.R. Hofstadter, "Gödel, Escher, Bach: Una eternal trenza dorada", CONACYT, México.

[8] "Breve Historia de la Inteligencia Artificial" en http://www.uned.es/pfp-internet-y-educacion/historia.html .

[9] Stuart Russell y Peter Norvig, "Inteligencia Artificial, Un enfoque moderno", Segunda edición, Pearson Education, S. A. Madrid, 2004, 1240 pp.

[10] Antonio Toca, "Wakamaru, un fiel compañero" en http://xataka.com/archivos/2005/02/26-wakamaru-un-fiel-companero.php

[11] Jesús Savage, Mark Billinghurst, Alistair Holden, "The Virbot: A virtual reality mobile robot driven with multimodal commands." En Expert Ssystems with Applications 15 (1998) pp 413-419.

[12] R. Shank, Conceptual Dependency: A theory of natural language understanding, *Cognitive Psychology 3*, 552-631 (1972).

[13] "What is CLIPS?" en http://www.ghg.net/clips/WhatIsCLIPS.html

[14] G. Hewer, C. Kenney, B.S. Manjunath, "Variational Image Segmentation using Boundary Functionals" en http://vision.ece.ucsb.edu/segmentation/variational/.

[15] Yining Deng, B.S. Manjunath, "JSEG - Segmentation of color-texture regions in images and video" en http://vision.ece.ucsb.edu/segmentation/jseg/.

[16] A. Ferencz, E. Learned-Miller, J. Malik, "Learning to Locate Informative Features for Visual Identification" en http://www.eecs.berkeley.edu/Research/Projects/CS/vision/shape/vid/ferencz-ijcvDRAFT.pdf, IJCV Special Issue, 2005.

[17] A. Ferencz, E.G. Learned-Miller, J. Malik, "Building a Classification Cascade for Visual Identification from One Example" en http://www.eecs.berkeley.edu/Research/Projects/CS/vision/shape/papers/ferencz-iccv05.pdf, ICCV 2005.

[18] Wolf Schaarschmidt, "Reconocimiento de gestos clave para interactuar con un robot móvil", tesis de maestría IIMAS, UNAM 2004.

[19] "Objeto", "Cosa" en Diccionario de la Lengua Española de la Real Academia. 22ª edición.

[20] Paul Graham "2.6 Closures" en On Lisp, pp 17.

[21] R. Molina, Introducción al Procesamiento y Análisis de Imágenes Digitales. Notas del curso "Introducción a la Robótica", Granada, España, 1998, 355 pp.

[22] Darío Cesar Peregrina Albores, "Seguimiento de Objetos por medio de Visión Activa", tesis de maestría INAOEP, 2002.

[23] Amoozegar Farid, "Neural-network-based target tracking state-of-the-art survey", Op. Engineering, Vol. 37, No. 3, SPIE Optical Engineering Press, 1998.

[24] Torres Méndez, Luz Abril, "Viterbi Algorithm in Text Recognition" http://linas.org/mirrors/www.cim.mcgill.ca/2002.09.19/~latorres/Viterbi/va_intro d.htm

[25] Sánchez Secades, Juan María, "Multiple feature temporal models for the characterization of semantic video contents", Tesis, Universidad Autónoma de Barcelona, 2003.