Agentes Inteligentes y Comunicación con Avatares

Jimena Olveres Montiel

Tesis para obtener el grado de Maestro en Ingeniería Eléctrica

Director de Tesis: Dr. Jesus Savage Carmona

octubre de 1999

A mi padre Arturo Olveres Roldán quien siempre me amó, apoyó y me ayudó a creer en mí.
Aunque hoy no te encuentres junto a mí tu recuerdo y enseñanzas siempre me acompañarán.

Índice General

Índice de Figuras

Resumen

En esta tesis se desarrollan un conjunto de programas que controlan agentes de software. Se incluyen dos enfoques de desarrollo: el control de acciones a ejecutar del sistema y el control de interfaces humanas basado en comportamientos emocionales. El objetivo fue contar con una base que sirva para aplicaciones que ejecuten órdenes y se tengan interfaces dentro de un entorno gráfico que realice el intercambio de ideas dinámico.

El primer enfoque ó control de acciones para agentes computacionales, instrumentó técnicas de Inteligencia Artificial que crea e identifica patrones de reconocimiento de acciones de un robot dentro de una casa. Para lograr esto se utilizan Dependencias Conceptuales que realizan reconocimiento y ejecución de guiones similares a los de teatro. El segundo desarrollo de interfaces emocionales facilitan la comunicación entre avatares. Avatares es una representación del usuario en un entorno gráfico utilizado a través de la red internet. Este proyecto se realizó utilizando sistemas expertos y modelos basados en funciones exponenciales y lógica difusa así como síntesis de voz. Esta interfaz ofreció la inferencia de emociones en un sistema de pláticas remotas. Las pruebas con usuarios se realizaron en la Universidad de Washington, Seattle y mostraron una serie de aspectos nuevos a considerar sobre el proyecto que no se habían contemplado al iniciar el presente trabajo, los cuales se presentan en la parte de resultados. En cuanto al sistema de control de acciones los guiones demostraron su funcionamiento óptimo al utilizarse en un ambiente simulado dentro de una casa.

El presente trabajo demuestra que la integración de diversas técnicas de "Inteligencia" mejora en gran manera la interacción hombre-máquina y permite el desarrollo de nuevas aplicaciones de interactividad avanzada, que ayudan a establecer la comunicación entre una o más personas en forma cotidiana.

Parte I

Estado del Arte

Capítulo 1

Introducción

1.1 Antecedentes

Mientras que las computadoras se tornan mas rápidas y con mayores capacidades gráficas, el número de aplicaciones aumenta constantemente. En estas aplicaciones la retroalimentación visual es muy importante y se desea que posean cada vez más facilidad y naturalidad en su uso. La mayoría de estas aplicaciones utilizan bibliotecas de animación a bajo nivel para controlar cada aspecto en la presentación. Mientras nuevas animaciones son creadas y se introducen nuevas configuraciones de escenas gráficas, se vuelve necesario realizar cambios significativos a la aplicación. Más aún la lógica para controlar estas animaciones se torna cada vez más compleja y dificil de codificar. Los guiones de animación o acciones por lo general poseen dependencias en las configuraciones de las escenas gráficas, y se introducen por razones semánticas o por razones de creación de nuevos guiones. Antes de invocar un guión se deben considerar que otros guiones se observen anteriormente. Y conforme cambien los detalles de una animación tal vez es necesario reconsiderar todas las etapas de transición dentro de estos guiones.

Por otro lado, con referencia a las aplicaciones computacionales, esta surgiendo un nuevo campo en el cual los programas actúan como asistentes en lugar de ser simples herramientas. Estas aplicaciones asumen responsabilidades para realizar subtareas cada vez más complejas. Con el proyecto Persona en Microsoft Research, se intentan desarrollar interfaces asistentes en la conversación por medio de caracteres de computadora que interactúan con los usuarios para ejecutar tareas y reportar resultados. Este proyecto intenta producir la ilusión de poseer un ser consciente dentro de la computadora. El usuario observa un ente en un espacio simulado, que

responde (verbalmente y ejecutando acciones) ante comandos de lenguaje natural. El énfasis principal que se da a estos agentes es en su capacidad de conversación y mímica para interactuar con un humano; para simular una "persona" y manejar el diálogo de manera imperceptible para cubrir las deficiencias de su reconocimiento y comprensión. Voz y procesamiento del lenguaje son elementos críticos pues al "hablar" es más fácil verlo como ente consciente. También se toma en cuenta la apariencia visual y los actos que realice para darle credibilidad al caracter. De igual manera un modelo de emociones del mismo otorga un mecanismo para adaptación gradual de su comportamiento [MICR]. Obviamente para obtener esto aún falta mucho desarrollo en el área, aunque el enfoque de la introducción de estos agentes de software es claro.

En el Laboratorio de Interfaces Inteligentes (LII) en el área de Ingeniería en Computación de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, se esta realizando un esfuerzo muy grande por contar con una infraestructura adecuada para realizar investigación en Agentes de Software. Un paso en esta dirección lo conformó el desarrollo de esta tesis que se realizó como parte de un convenio entre la Universidad de Washington y el LII. La parte de comunicación con avatares fue propuesta por el Human Interface Technology Laboratory (HITL) de esta misma universidad, bajo la dirección del Dr. Alistair Holden y el M.S. Mark Billinghurst del departamento de Ingeniería Eléctrica. Las pruebas del sistema se realizaron en el HITL con personas a las cuales se convocó a participar en el proyecto.

El primer proyecto propuesto, consistió en dotar a un robot de la capacidad para ejecutar una serie de acciones utilizando guiones que implican almacenamiento del conocimiento. Al mismo tiempo también se propuso la creación de un agente conversacional que sea capaz de realizar despliegues emocionales, de manera que el usuario llegue a considerarlo casi humano.

1.2 Definición del problema de Tesis

Implantar mecanismos de seguimiento de acciones, así como de desición de despliegues faciales para obtener agentes de software con la suficiente apariencia humana, basado en lógica difusa, dependencias conceptuales y guiónes para el proceso de la información y ejecución de órdenes.

1.3 Objetivo

Contar con agentes de software que ejecuten órdenes de acuerdo a acciones predeterminadas por un guión. Y desarrollar agentes que emulen el comportamiento humano dentro de la comunicación, con cambios de estados emocionales dentro de un entorno estático. Estos agentes de software se utilizarán para crear sistemas de cómputo más amigables.

1.4 Hipótesis

Al iniciar este trabajo existieron varias hipótesis a verificar:

- Los guiones facilitan el reconocimiento y ejecución de acciones y proveen de autonomía a los agentes al mostrarles las diferentes etapas que conforman estas acciones.
- El uso de los guiones facilita implantar un robot móvil que opere en un entorno simulado dentro de una casa.
- Los sistemas expertos son aplicables para resolver problemas de toma de decisiones y ejecución de acciones en agentes computacionales.
- El uso de sistemas expertos combinados con técnicas simples de lógica difusa es aplicable en el proceso de humanizar estos agentes computacionales.

1.5 Limitaciones

Las principales limitaciones se encuentran en la implementación del software necesario para desarrollar el sistema, en cuanto a los despliegues emocionales nos encontramos con entradas solo texto de los cuales se quiere obtener toda la información emocional posible, así como las órdenes necesarias para el control de los guiones. Siendo que también existe otro tipo de entradas para obtener este tipo de información como por ejemplo el reconocimiento de gestos o imágenes.

En cuanto al robot, es necesario establecer una manera de implementar los guiones pertenecientes a acciones complejas, de manera que también se tenga la posibilidad de aprendizaje por parte del robot al observar operaciones comunes y repetitivas.

1.6 Alcances

Este proyecto tuvo como propósito principal obtener agentes de software a los que se les pudiesen agregar módulos de comportamiento y "conocimientos" que simularan inteligencia. Razón por la cual se realizaron dos proyectos simultáneos que posteriormente puedan unirse en uno solo. El primero fue el del robot navegando y ayudando dentro de una casa y el segundo el de una interfase emocional inteligente.

Aunque los resultados se aplicarán directamente al equipo con el que se cuentó en el LII, se espera que se puedan implantar en cualquier otra plataforma de cómputo similares al del LII sin necesidad de realizar cambios importantes.

El proyecto se consideró terminado en cuanto se contó con los dos agentes (el de reconocimiento y ejecución de acciones automáticas, y el de despliegues emocionales autónomos) funcionando correctamente.

1.7 Justificación

Los agentes de software es una disciplina que emerge de la conjunción de conocimientos de Ingeniería Eléctrica, Ingeniería en Computación y Ciencias de la Computación. Estos agentes poseen un conjunto de programas, escritos en software, que determinan su comportamiento.

Este trabajo cubre principalmente la interpretación de la información que se recibe textualmente de la computadora y el comportamiento de entes computacionales que ya cuentan con una interfaz adecuada para realizar acciones y observar comportamiento autónomo.

La principal motivación para realizar este trabajo fué: utilizar y crear sistemas expertos aplicados a agentes de software que operen en un entorno computacional utilizando el enfoque del comportamiento de entes inteligentes y técnicas de inteligencia artificial.

En el capítulo 1 se muestra una breve introducción sobre el tema que gira el presente trabajo así como las posibles hipótesis que se tuvieron al iniciar.

En los dos siguientes capítulos 2 y 3 se intenta dar una breve introducción a la teoría de agentes computacionales así como la de sistemas expertos con el fin de dar las bases necesarias y justificar en forma más amplia este trabajo.

En el capítulo 4 se muestra una reseña de la complejidad que posee el procesamiento del lenguaje natural y técnicas que muestran como se ha intentado resolver este problema.

El capítulo 5 y 6 habla del manejo e importancia psicológica que juegan las

emociones dentro de la comunicación humana, y que se han utilizado dentro de los entornos visuales de computadora.

En los siguientes capítulos 7 y 8 se muestra la manera en que se resolvieron las hipótesis a comprobar utilizando técnicas basadas en los capanteriores, así como los resultados a los que se llegó.

El capítulo 9 muestra los resultados obtenidas de las aplicaciones creadas. Por último las conclusiones del trabajo conforman el capítulo 10.

Capítulo 2

Sistemas Expertos

A medida que comenzaron las investigaciones en el área del procesamiento de lenguaje natural los investigadores observaron que los sistemas necesitaban una base teórica para continuar con su desarrollo. Desde el punto de vista del departamento de Computación y Psicología de la Universidad de Yale, Inteligencia Artificial (IA) es el conocimiento cuyo modelo es el comportamiento de aquellas personas a las que se les considera inteligentes. Un sistema computacional intenta simular este comportamiento. A pesar de tener esta finalidad en los programas realizados dentro de esta línea, no necesariamente son el mejor modelo humano y por lo tanto siempre se encuentran sujetos a fallas [SCHA81].

La meta final de la IA es hacer máquinas inteligentes y su definición se orienta hacia el lado de la simulación cognositiva. En general la IA se enfoca en el tipo de conocimiento que posee la gente, la manera en que lo almacena, accesa, aplica y adquiere. Este enfoque resulta al comprender que el conocimiento humano está más alla de la habilidad de las personas.

Las técnicas de inteligencia artificial siempre han presentado ciertos retos como son el manejo de problemas para representar el conocimiento. Los sistemas de IA han manejado este conocimiento a través de dos formas. La primera es acoplar el desempeño humano en el mundo diario particularmente para comprender el lenguaje natural. Y la segunda se hizo al crear sistemas expertos llamados también sistemas basados en conocimiento que pudieran acoplar la experiencia humana en tareas bien definidas [SCHA81]. Los primeros sistemas expertos se enfocan a sistemas de pregunta respuesta en lugar de sistemas que ejecuten decisiones.

El primer paso al resolver cualquier problema es definir el área o dominio que éste ocupa. Esta consideración se utiliza tanto en el área de IA como en cualquier área de

programación convencional. Aunque aun no se han encontrado soluciones completas a problemas clásicos como son interpretación del lenguaje natural, comprensión del lenguaje y visión, restringir el dominio del problema otorga soluciones aceptables. Por ejemplo no es difícil construir sistemas sencillos del lenguaje natural si su entrada se restringe a enunciados de la forma sujeto, verbo y complemento. Actualmente los sistemos de este tipo funcionan adecuadamente al combinarse con interfaces de usuarios amistosa. De hecho los sistemas de procesamiento asociados con juegos de computadora que manejan texto poseen una gran habilidad de entendimiento del lenguaje natural.

En la Figura 2.1 se muestra el enfoque que da Giarratano [GIAR94] sobre las áreas de interés de IA: robótica, visión, voz, sistemas neuronales, comprensión, lenguaje natural y sistemas expertos. El área de los sistemas expertos es de las más explotadas en forma exitosa por dar soluciones a los problemas clásicos.

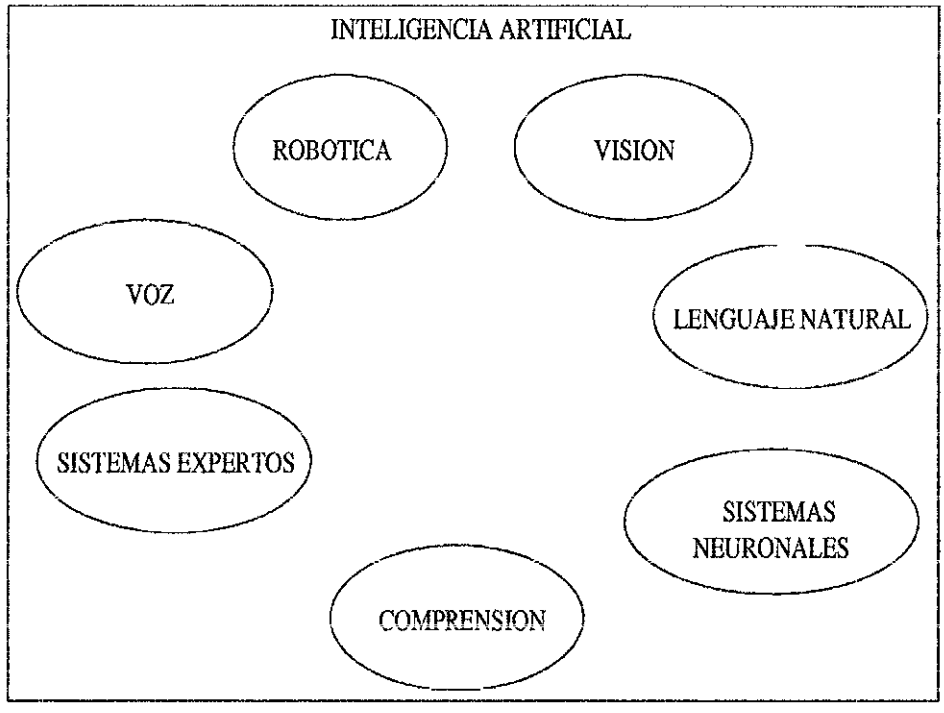

Figura 2.1: **Áreas de interés para la Inteligencia Artificial.**

2.1 Sistema Experto

El profesor Edward Feigenbaum de la Universidad de Stanford define un sistema experto (SE) como *un programa de computadora inteligente que usa el conocimiento e infiere procedimientos para resolver problemas que son los suficientemente complicados de manera que requieran de un experto humano para su solución.* Es decir, un SE es un sistema de computación que emula la toma de decisiones como lo hace un experto humano. El término emular significa que el SE intenta actuar en todo respecto de manera similar al humano.

Cuando los SE fueron desarrollados en los 70's únicamente se enfocaban a bases de datos con conocimiento experto, sin embargo se utiliza el mismo término para cualquier sistema que utiliza las tecnologías mencionadas de los SE. Esta tecnología puede incluir tanto lenguajes especiales para el diseño de los programas así como hardware diseñado para mejorar la ejecución del sistema.

Dentro del concepto básico de un sistema experto [GIAR94], el usuario proporciona hechos u otra información al SE y este le da una respuesta en contestación. Inicialmente se pueden reconocer dos componentes principales: El conocimiento base es el conocimiento que la máquina de inferencias utiliza para obtener conclusiones. Estas conclusiones son las respuestas que se proporcionarán como resultado. Se denomina dominio al conocimiento del experto, utilizado para resolver problemas.

Para que un SE resuelva problemas como normalmente lo haría un experto humano, se requiere el acceso a una base de datos sustancial, la cual se debe de construir lo más eficientemente posible. De igual manera deben de poseer mecanismos de razonamiento para aplicar este conocimiento. También necesitan de un mecanismo que explique lo que se realizó. Los problemas con los que se enfrentan los SE son altamente diversos, en sí son también conocidos como programas complejos de inteligencia.

Newell and Simon en su libro Human Problem Solving [NEWE72] explican que la mayor parte de la resolución de problemas o conocimiento puede expresarse en forma de reglas de producción tipo **SI-ENTONCES** (*IF-THEN*). Cada regla corresponde a una colección modular de conocimiento denominadas bloques. Estos bloques se organizan en un arreglo que contienen enlaces a otros bloques de conocimiento.

El conocimiento codificado en reglas puede ser heurístico. Este término tiene raíces griegas y significa descubrir, y se trata de un método que permite la solución de un problema. Sin embargo un método heurístico no garantiza encontrar la solución, esto sólo lo puede lograr un algoritmo. En un programa convencional, de forma

secuencial, usualmente no se tiene una buena idea del algoritmo al momento de iniciar el programa. En un SE en cambio sucede que no se tiene un buen algoritmo frecuentemente, debido a que el conocimiento es incompleto o heurístico. Cada regla se forma por un extremo izquierdo que necesita ser satisfecho (Hechos) y por un extremo derecho que produce la respuesta apropriada (Acciones)

SI **Hechos** ENTONCES **Acciones.**

Por ejemplo:
Si la luz es **roja** *entonces* **deténgase**.
La regla es satisfecha y realiza la acción de alto. Aunque este es un ejemplo muy sencillo, la mayoría de los SEs se han construido de esta manera y llegan a ser bastante complejos conforme el código aumenta. Inclusive algunas herramientas de SEs como son **CLIPS** pueden permitir uso de objetos tanto como reglas para almacenar el conocimiento.

La manera en que las reglas expiden una acción es creando hechos de activación. Por ejemplo el hecho creado de la regla anterior es: *orden de detenerse.* Y este hecho activa otra regla que realiza la acción. De esta manera una acción completa se constituye de enlaces a otras reglas. Puede suceder que un programa utilice miles de reglas para resolver un problema por lo que necesita un mecanismo especial que seleccione las reglas que serán disparadas de acuerdo a los hechos presentes. Ese mecanismo es denominado Máquina de Inferencias del SE. Los elementos típicos de un sistema experto de acuerdo a Giarrratano y Riley [GIAR94] [SAVA95] se muestran en la Figura 2.2:

- **Interfaz de usuario:** Los medios por los cuales el sistema experto y los usuarios se comunican.
- **Manejador de razonamiento:** Se encarga de explicar al usuario la manera en que se razonan las conclusiones.
- **Memoria utilizada:** Una base de datos global donde se almacenan todos los hechos utilizados por las reglas.
- **Máquina de Inferencia:** Decide o infiere que reglas son satisfechas por los hechos dando prioridad a las reglas a satisfacer, y ejecutando la regla con la más alta prioridad.

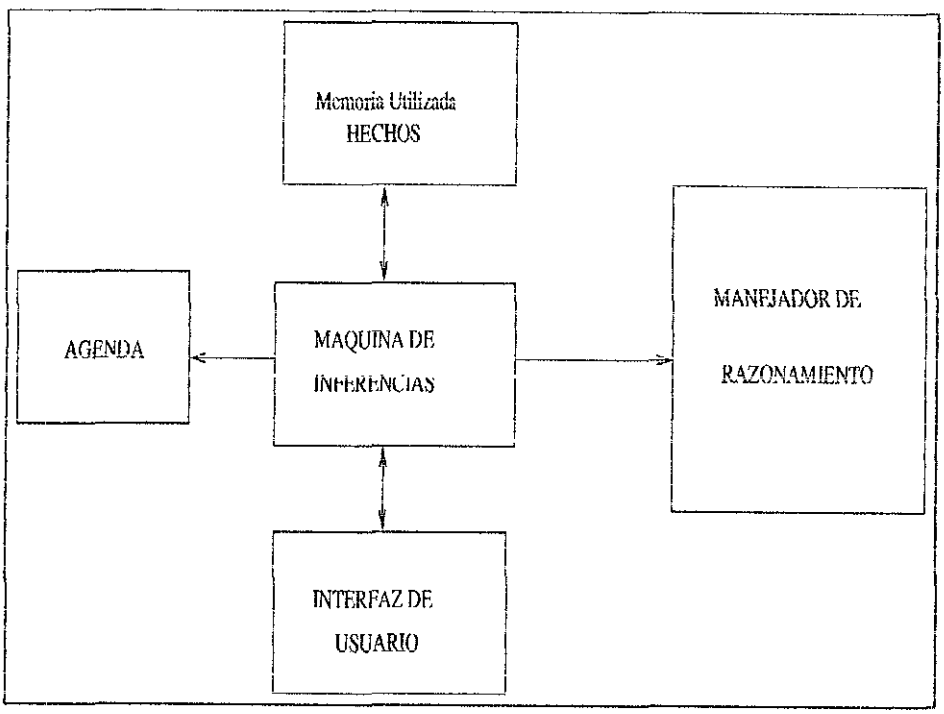

Figura 2.2: **Componentes de un Sistema Experto.**

- **Agenda:** Es una lista de reglas ordenadas segun prioridades otorgadas por la máquina de inferencias cuyos patrones se satisfecen por los hechos en la memoria utilizada.

- **Manejador de Adquisición de datos:** Da la pauta para introducir conocimiento nuevo sin la necesidad de codificarlo.

La parte de adquisición del conocimiento es una característica opcional en muchos sistemas. La base de datos del conocimiento es también llamada memoria de producción. En un sistema basado en reglas la máquina de inferencias determina que regla puede ser satisfecha por los hechos presentes. Existen dos métodos de inferencia comunmente usados: Encadenamiento hacia adelante y Encadenamiento hacia atrás. Encadenamiento hacia adelante es razonar las conclusiones resultantes partiendo de los hechos presentes. Por ejemplo:

Hecho: si se ve que esta lloviendo antes de dejar la casa,

Conclusión: entonces lleve un paraguas.

Encadenamiento hacia atrás incluye razonamiento en sentido inverso, es decir de una hipótesis supone una conclusión potencial que aún no ha ocurrido, debido a los hechos que soportan esa hipótesis. Por ejemplo si el sujeto no se ha asomado a la ventana pero alguien entra con los zapatos mojados y un paraguas, lo más razonable es que este lloviendo. Para darle validez a esta hipótesis se puede preguntar a la persona si esta lloviendo. Esta hipótesis se puede manejar como una meta a ser alcanzada. Una regla cuyos patrones son satisfechos se le llama activada o instanciada. Múltiples reglas pueden ser activadas en la agenda al mismo tiempo. En este caso la máquina de inferencias debe seleccionar una regla para disparar. El término disparo proviene de la neurofisiología, ciencia que se dedica al estudio del sistema nervioso. Una célula o neurona emite una señal eléctrica cuando se estimula y en este momento se inhibe a cualquier otro estímulo que cause un nuevo disparo de la neurona por un periodo de tiempo. Este fenómeno es llamado refracción. Los sistemas expertos basados en reglas se construyen usando refracción para evitar ciclos o repeticiones innecesarios.

Como se dijo entonces la máquina de inferencias necesita encontrar que reglas son satisfechas por los hechos y ejecutarlas, pero si el sistema tiene miles de reglas este proceso requiere mucho tiempo de procesamiento. Una solución a este problema es la que da el algoritmo Rete, desarrollado por Charles Forgy [FORG82]. La idea de este algoritmo es que en lugar de buscar los hechos que satisfacen las reglas, los hechos son los que buscan las propias reglas (vease Figura 2.3). De manera que

cuando un nuevo hecho se crea la máquina de inferencias checará si es que activa o no alguna regla.

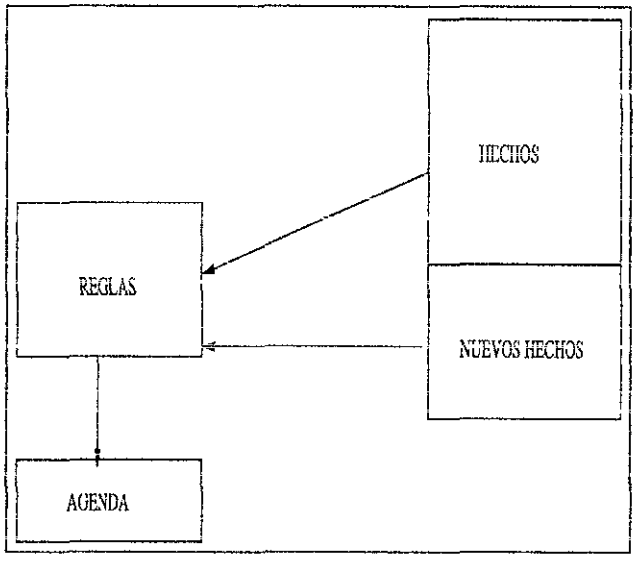

Figura 2.3: **En el Algoritmo Rete los nuevos hechos buscan las reglas.**

Las reglas son decodificadas para formar una red (Rete en italiano), en la cual los nodos de la red son reglas que comparten los mismos hechos. Esta red almacena información que poseen las reglas. Se utiliza una lista de reglas que son preactivadas por los hechos presentes, y debido a que el número de nuevos hechos es pequeño durante un ciclo de ejecución, es más rápido al checar si estos nuevos hechos satisfacen el lado izquierdo de las reglas preactivadas.

En la parte de aplicaciones se muestra la manera en que se utilizaron y desarrollaron sistemas expertos en el presente trabajo.

Capítulo 3

Agentes

Desde mucho tiempo atrás las personas se encuentran fascinadas con la idea de los agentes no humanos. Las historias populares de androides, robots, humanoides y ciencia ficción siempre han llamado la atención. Pero a pesar de éstos comienzos la imagen pública de seres artificialmente inteligentes son por lo general más que una pesadilla un sueño. Sin embargo el deseo de que un robot exhiba características más humanas es latente y ha dado lugar al desarrollo de agentes, especialmente aquellos dentro del área de software. La noción de agente ha sido de gran importancia dentro de la investigación de la Inteligencia Artificial Distribuida (AID). El término agente puede incluir desde un sistema biológico primitivo hasta un equipo de alta tecnología como puede ser un robot o avión, e inclusive sistemas que simulen o describan sociedades humanas.

Un modelo general reducido de un agente se muestra en el sistema típico que se representa por una caja negra, Figura 3.1 [MULL96]. El sistema se describe internamente por una función f que percibe y recibe mensajes como entrada para después generar una salida en términos de sus entradas. El mapeo f no se encuentra controlado por una autoridad externa por lo cual siempre se considera autónomo. Este modelo puede aplicarse a una serie de procesos, sin embargo la diferencia estriba en la naturaleza de la función f, la cual determina el comportamiento del agente . Desde el punto de vista del diseñador de sistemas, un agente siempre tiene adscrito un propósito.

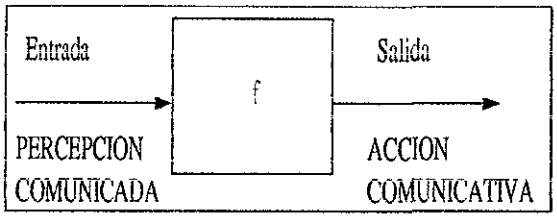

Figura 3.1: **Modelo simple de un sistema.**

3.1 Disciplinas relacionadas

Al menos tres áreas principales han sido relacionadas con los agentes: la teoría de control, la psicología cognosítiva y la planeación perteneciente al área de la IA [BRAD97].

La teoría de control clásica proporciona un modelo matemático para describir la interacción de un sistema dinámico, en el cual se tiene un controlador y un medio ambiente a controlar. Sin embargo la diferencia que existe con respecto a un agente es que la función f se representa con ecuaciones de control, mientras que el otro utiliza razonamiento simbólico. Desde el punto de vista del control, muchos desarrollos de planeación se basan en sistemas de alimentación hacia adelante debido a que esos predicen la reacción del ambiente frente a ciertas acciones de los agentes, y estas acciones crean situaciones hipotéticas que planean el curso de acción óptimo. Este último comportamiento se atribuye a los sistemas reactivos, en los que el siguiente paso a ejecutar se determina por un conjunto de reglas situadas bajo ciertas precondiciones satisfechas.

La psicología cognitiva comprende aquellas metas e intenciones de un agente, que dirigen la ejecución de actos que cambian el estado del mundo. En particular comprende la teoría de la motivación. Esta teoría trata de responder el porque un agente realiza cierta acción o revela cierto comportamiento. Por lo cual cubre la transición de la motivación a la acción. Se tiene énfasis en dos aspectos importantes: la formación de la intención al empezar con un conjunto de motivaciones sobre el cual se observará una cierta tendencia, y segundo cómo es que estas intenciones se ponen en práctica. Es decir, el porque, cómo y cuándo se inicia la acción.

Los sistemas de inteligencia artificial como son STRIPS [HUTC95] ven el comportamiento de los agentes como un acto cíclico de planeación. Se compone del estado actual del medio, el estado final y un conjunto de operaciones. Dada esta descripción el problema es encontrar una secuencia de ejecuciones que vayan del

estado inicial al estado final. Por esta razón la planeación implica una búsqueda dentro de un conjunto de estados, de los cuales se elige el que más factiblemente encamine el agente al estado final.

En la siguiente sección se trata de dar una definicíon de agente y como toma elementos correspondientes a estas áreas afines.

3.2 Definición y clasificaciones

Un agente es cualquier ente que puede ser visto como un receptor de su ambiente a través de sensores y actuando sobre ese ambiente por medio de ciertos efectos. Un agente humano tiene ojos, oidos y otros órganos como sensores, manos piernas, boca y otras partes corporales que modifican el ambiente [DAME98].

Una definición más específica de agentes de software, que muchos investigadores consideran aceptable, es aquella entidad computacional que funciona contínuamente de una manera autonóma en un ambiente particular, por lo general habitado por otros agentes y procesos. El requerimento de continuidad y autonomía deriva del deseo de que el agente pueda llevar a cabo actividades en forma inteligente y autónoma para que responda a cambios en el ambiente sin requerir intervención y guía humana [BRAD97].

Idealmente un agente que funciona en un ambiente de manera continua tendrá la capacidad de aprender de su experiencia. Además de convivir con otros agentes debe ser capaz de comunicarse y cooperar con ellos. Hoy día la cobertura del término es muy amplia e incluye a otros agentes más específicos y limitados.

Dependiendo de los requerimentos de un problema particular cada agente puede poseer, en menor o mayor grado, atributos tales como:

- **Reactividad:** la habilidad para sensar y actuar selectivamente.
- **Autonomía:** que le dirige al cumplimiento de sus metas, a ser proactivo y a tener un comportamiento individual.
- **Comportamiento colaborativo:** que le permite trabajar junto con otros agentes para lograr una meta común.
- **Nivel de Conocimiento:** como por ejemplo la habilidad para comunicarse con personas y otros agentes utilizando lenguaje natural, en lugar de protocolos que utilizan sólo símbolos.

- **Capacidad de inferencia:** que le facilite abstraer la especificación de tareas usando un conocimiento previo de metas y métodos generales que le otorguen flexibilida, es decir un amplia gama de comportamientos, al actuar. Esta característica va más allá de la información dada y utiliza modelos específicos de comportamiento.

- **Continuidad:** permite que permanezca la identidad y el estado en largos períodos de tiempo.

- **Personalidad:** es la capacidad de manifestar los atributos de un caracter, como por ejemplo las emociones.

- **Adaptación:** capacidad de aprendizaje y mejoramiento.

- **Movilidad:** Capacidad propia del agente de migrar de un lugar a otro con cierta dirección.

Un agente se constituirá de una combinación de estos posibles atributos aunque no necesariamente presenta todos.

De la comunidad IAD también se ha llegado a una clasificación muy difundida la cual caracteriza los agentes de acuerdo al grado que manejan para resolver problemas: Agentes Deliberativos, Agentes Reactivos, Agentes Interactivos y Agentes Híbridos [MULL96].

Los Agentes Deliberativos son aquellos que poseen una representación interna de su mundo y poseen un estado mental que se modifica por alguna forma de razonamiento simbólico. Este modelo se conoce como arquitectura de Creencia, Deseo en Intención (CDI).

La idea básica es describir el estado interno del agente a través de un conjunto de categorías mentales y definir una arquitectura de control mediante la cual el agente seleccione "racionalmente" su curso de acción. Las categorías mentales son creencias o conocimientos de sí mismo, deseos e intenciones, y en la práctica estas categorías se completan con las nociones de fin último y planeación.

Mediante sus creencias, un agente expresa sus expectaciones acerca del estado del mundo actual, de igual forma muestra la semántica de mundos posibles en los que puede creer. Deseo es una noción abstracta que especifica preferencias sobre estados del mundo futuro o cursos de acción. Debe permitírsele al agente poseer deseos inconsistentes los cuales sepa que no son posibles, es decir sueños. Las metas

se establecen una vez que ya se tiene un conjunto finito de deseos que el agente puede realizar, sin embargo aún no se toma el curso específico de acción. Esto se hará mediante una transición de metas a intenciones y adicionalmente se requiere que el agente crea realizables sus metas. Un agente no puede realizar todas sus metas en un mismo tiempo. Aún si éstas son consistentes es necesario seleccionar una sola o disminuir su conjunto. Por último aunque no es ingrediente conceptual en la teoria de CDI, la planeación permite el desarrollo pragmático de las intenciones.

Los Agentes Reactivos toman sus decisiones al momento de correr el sistema. Usualmente se basa en una limitada cantidad de información y reglas que se disparan de acuerdo a la situación. Su comportamiento varía ante cambios del ambiente o ante mensajes transmitidos por otros agentes. Estos agentes toman las decisiones basándose en entradas sensoriales, lo cual origina que el sistema sea más robusto.

Agentes Interactivos o sociales, la IAD trabaja en la coordinación entre agentes inteligentes distribuidos y se relacionan con el proceso de coordinación y creación de mecanismos de cooperación entre agentes autónomos. De los principales temas que estos agentes abarcan se encuentran: comunicación, que es el intercambio de información y conocimiento para lograr una mejor interacción utilizando voz en las etapas de planeación; planeación multiagentes relacionada a la resolución de problemas distribuidos y la generación y ejecución de planes por múltiples agentes; y por último resolución de conflictos y cooperación a través de negociaciones. Todos estos mecanismos son aquellos que no se encargan de describir al agente como individuo.

Los Agentes Híbridos integran todos los anteriormente mencionados en un solo agente, y una de las formas que propone es a través de etapas o niveles. Entre las ventajas que un sistema de etapas ofrece se encuentran:

- modularización de un agente con sus funciones claramente delimitadas y con comunicación definida;
- hace que el diseño sea más compacto;
- incrementa la robustez y facilita el encontrar errores;
- como diferentes etapas pueden funcionar en paralelo, la capacidad computacional del agente se incrementa en forma lineal, especialmente en la reactividad. Por ejemplo cuando un agente se encuentra en medio de una planeación, otra de las etapas reactivas aún puede monitorear el medio para situaciones imprevistas.

Existen además características especifícas que conforman a los agentes inteligentes. Un famoso artículo de IBM [GILB95] describe agentes inteligentes en términos de un espacio definido por tres dimensiones: agencia, inteligencia y movilidad (ver Figura 3.2)

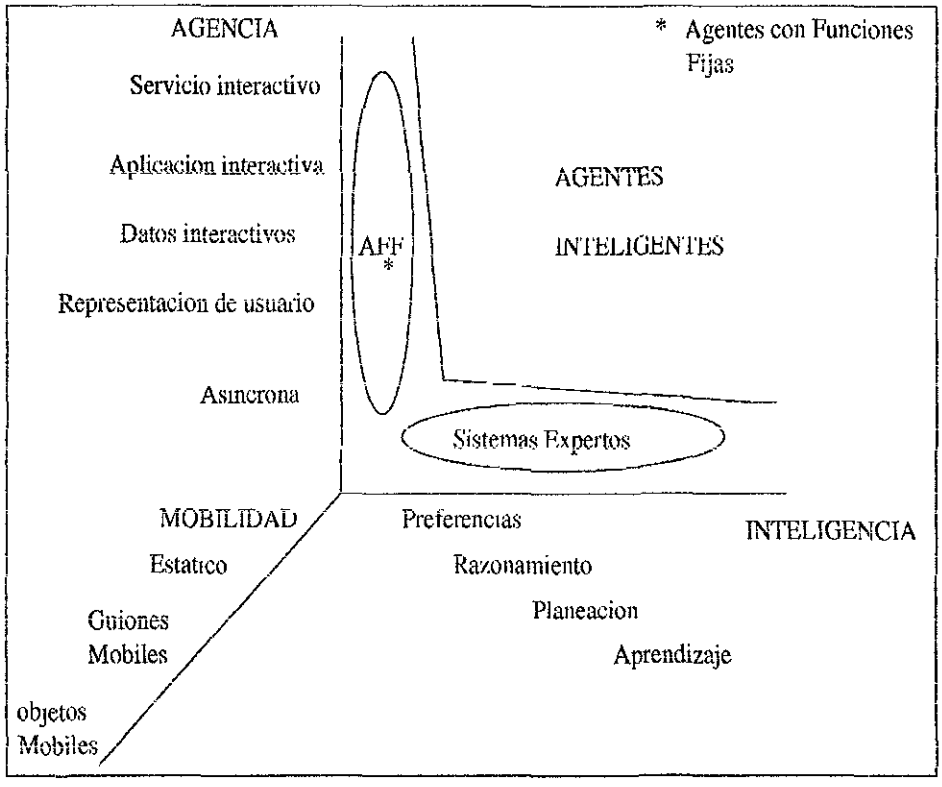

Figura 3.2: **Dimensiones de un Agente.**

- **Agencia** es el grado de autonomía y autoridad investida en el agente y puede ser medida, al menos cualitativamente, por la naturaleza de la interacción entre el agente y otras entidades en el sistema. El grado de agencia se puede analizar mejor si el agente representa un usuario (*avatar*), y si es más avanzado puede interactuar con datos, aplicaciones, servicios u otros agentes.

- **Inteligencia** es el grado de razonamiento y aprendizaje, es la habilidad que se tiene para realizar las tareas otorgadas y llegar a la meta especificada. Al menos debe de existir alguna jerarquización de acciones preferentes a realizar.

Modelos más sofisticados incluyen razonamiento y sistemas que aprenden y se adaptan a su ambiente.

- **Movilidad** es el grado con el que viajan los agentes, por ejemplo a través del Internet. Un ejemplo específico es el de los guiónes móviles, que se crean en una máquina y se envían a otra para su ejecución, los objetos móviles son transportados de máquina en máquina en el medio de una ejecución y llevan consigo datos acumulados.

Algunas otras clasificaciones incluyen otros aspectos extras a los ya mencionados como son el tipo de información que manejan y atributos secundarios como versatilidad, veracidad, así como características emocionales y mentales.

Se puede definir entonces como agente autónomo a un sistema situado dentro de un ambiente que sensa su estado en el tiempo, y actúa sobre él modificando su propia agenda en la que lleva los posibles efectos en el futuro. A continuación se muestra en la Figura 3.3, una de las taxonomías que cubren de manera más amplia los ejemplos encontrados en la literatura según Bradshaw.

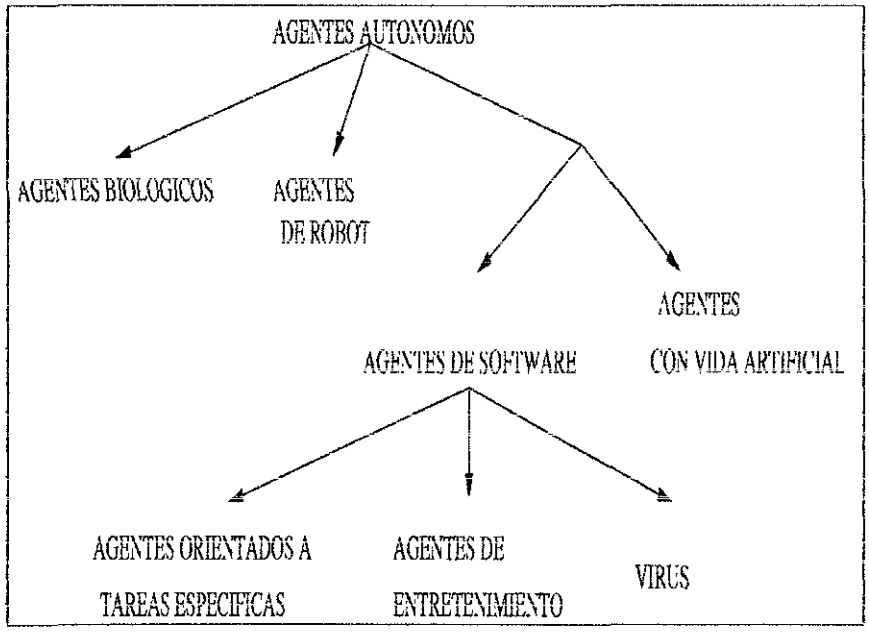

Figura 3.3: **Modelo Taxonómico de Agentes.**

El trabajo de la inteligencia artificial es diseñar el control del agente, una función que haga el mapeo para realizar la acción que se le pide. Este programa corre sobre una base llamada arquitectura en la cual cada agente usa estructuras de datos internas que son actualizadas según aparecen nuevos estímulos. Estas estructuras son operadas mediante decisiones del mismo y generan acciones a ejecutarse.

3.2.1 Avatares

Un término común de agentes utilizado en los ambientes gráficos del Internet es el de avatares:

Avatares son representaciones digitales de los usuarios dentro del internet, usualmente caracteres gráficos que permite explorar dentro de los ambientes gáficos computacionales. Un ejemplo muy de moda en nuestros días son los mundos virtuales, los cuales pueden representar el centro de una ciudad, un parque, tiraderos de basura o ambientes de fantasia con una arquitectura flotante y con formas de locomoción que rompen cada ley de la física [DAME98], y en los cuales el usuario puede navegar como si se encontrara dentro del mismo. El primero en utilizar el término *avatar* fue Chip Morningstar, al describir cuerpos visuales en el ambiente *Habitat* en 1985. También se denominan caracteres, actores, iconos o humanos virtuales.

3.3 Modelo Conceptual

Un agente autónomo que trabaja en un ambiente dinámico multiagente tiene que tomar en cuenta requerimientos de tiempo real y del mundo real. Se escogió como base a seguir el modelo conceptual desarrollado por Muller [MULL96], por ser uno de los más completos. En el que se trata de cubrir características basadas en niveles. Estos aspectos definen al sistema **INTERRAP** que se conforma de tres niveles principales:

- Un nivel de comportamiento que incorpore reactividad ofreciendo un comportamiento según situaciones, y conocimiento procedural para tareas de rutina.
- Un nivel de planeación local que satisfaga un razonamiento dirigido al cumplimiento de las metas que tiene el agente, otorgando eficiencia.
- Un nivel de planeación cooperativa que permita el razonar acerca de los demás agentes y coordine acciones conjuntas.

Para lograr esto el sistema integra diferentes funciones de abstracción, que conforman a INTERRAP como un sistema CDI con entradas (percibidas de su entorno) y salidas (o acciones). El estado mental de los agentes se compone de:

- La percepción actual del agente.
- Un conjunto de creencias que denotan su conocimiento.
- Un conjunto de situaciones que muestran componentes relevantes dentro de estas creencias.
- Un conjunto de metas que el agente aspira a tener.
- Un conjunto de intenciones que definan, en base a lo que ejecutado, la acción siguiente.
- Un conjunto de primitivas que hagan que el agente sea capaz de deliberar.

En la Figura 3.4 se observa el diagrama del modelo conceptual. Las creencias se dividen en creencias del modelo ambiental, creencias del modelo mental y creencias del modelo social. El modelo mental contiene todo aquel conocimiento del agente sobre sí mismo. El modelo ambiental le otorga el conocimiento para interactuar con su medio. Y el social es el conocimiento acerca de los demás agentes.

A partir de este modelo se genera un reconocimiento de situaciones que se presenten. Las situaciones específicas activan una cierta acción. Estas situaciones representan estados ambientales que interesan al agente, y se distinguen tres tipos de situaciones. Las situaciones de rutina, las situaciones que exigen una planeación local basádas en un modelo ambiental y un modelo mental propio. Y las situaciones que requieren planeación cooperativa y que son parte del mundo social del agente.

De igual manera a partir de estas situaciones que surgieron se establecerán metas que el agente va a ejecutar. Las metas se clasifican en: metas locales, metas cooperativas y metas de reacción que son a corto plazo, por lo general y se activan por eventos externos que requieren una reacción inmediata.

Las Primitivas Operacionales (PO) permiten el razonamiento para obtener ciertas metas, estas POs denotan también patrones de comportamiento al presentarse situaciones comunes y están relacionados directamente con las metas presentes del agente. Toda esta estructura (los modelos, situaciones y primitivas operacionales) viene a generar el conjunto de intenciones del agente.

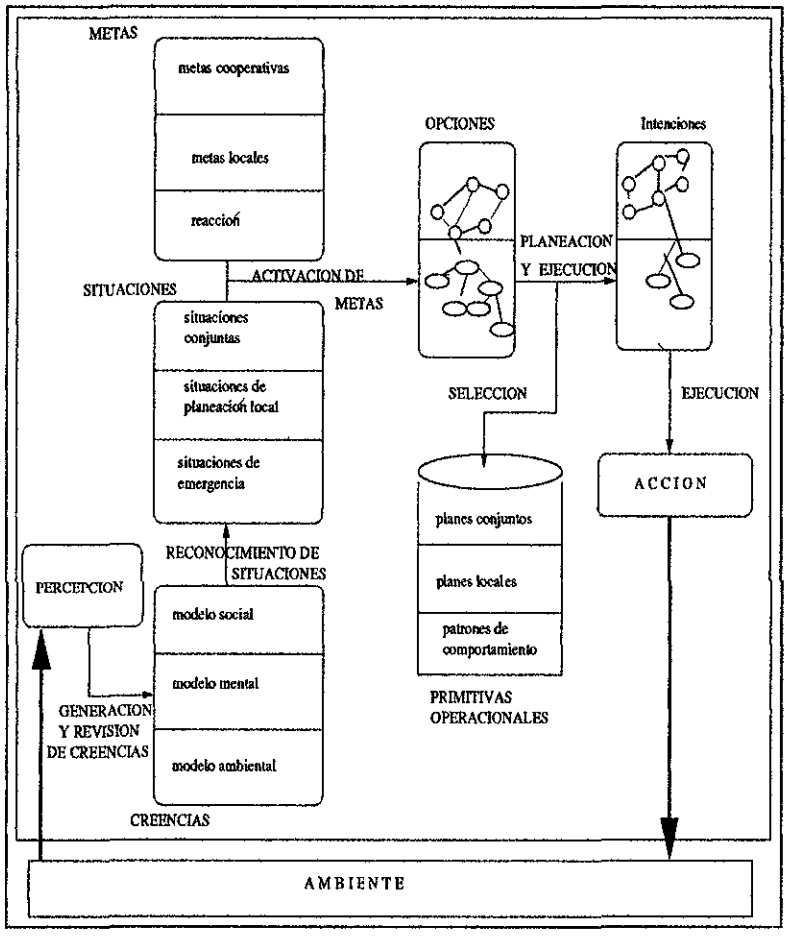

Figura 3.4: **Modelo Conceptual de INTERRAP [MULL96].**

La dirección de flujo que se presenta en la Figura 3.4 representa la relación funcional entre los componentes del modelo mental del agente y muestra como se mapean las percepciones en acciones. Estas relaciones funcionales son:

- **Generación y revisión de creencias.** Establece la relación entre las percepciones del agente y sus creencias. La representación de estas percepciones se hace en forma simbólica al igual que la de las creencias.
- **Reconocimiento de Situaciones.** extrae situaciones estructuradas de una serie de conocimientos no estructurados del agente que permiten identificar la necesidad de actividad.
- **Activación de metas.** Describe que metas posibles se tienen con las condiciones presentes.
- **Planeación.** Mapea las metas en primitivas operacionales, es decir, decide los actos del agente.
- **Temporización.** Es el proceso mediante el cual se les otorga un orden de ejecución a cada plan local tomando en cuenta que procesos preceden a otros.
- **Ejecución.** Es responsable del desarrollo correcto, con un orden de labores determinadas por la planeación y temporización.

Relacionando este concepto con la investigación realizada tenemos lo siguiente. El sistema del robot navegando dentro de una casa toma en cuenta los elementos presentados en el modelo de Muller. En cuanto a la generación y revisión de creencias no se consideraron ninguna de las categorías de los modelos mentales. En su lugar se tomó un modelo general, el cual ve las percepciones del robot a través de representaciones simbólicas. Tal es el caso de una representación simbólica para la casa, una representación simbólica para los objetos encontrados dentro de ella (como son la silla), y otra representación para otros agentes también dentro de la casa.

Para el reconocimiento de situaciones se trató de crear un sistema eficiente y con reacción inmediata. Este intentará reconocer situaciones de emergencia y situaciones de planeación local. Sin embargo el número de situaciones a contemplar en ambos niveles puede ser inacabable, por lo cual se han delimitado a funciones de ayuda dentro de una casa en el caso del robot. Y a un cierto número de emociones presentes dentro de un texto en el caso de la interfaz de conversación. Es lógico suponer que si se cuenta con más tiempo es posible observar los posible efectos que las metas locales de un agente crean en las metas de los demás.

La activación de metas se realiza después de obtener la comprensión semantica que se necesitaba, es decir una vez que se ha reconocido adecuadamente la situación presentada ante el agente. Su funcionamiento se puede comparar al de una regla en un sistema experto, en la cual al cumplirse ciertas condiciones activan una acción. Sin embargo en la planeación local y cooperativa este proceso de activación de metas es mucho más complejo, pues la ejecución de una meta es solo un paso para construir una meta final. Esta etapa se desarrolló para el robot dentro de la casa.

La función de planeación como se dijo es especificar que hacer para obtener la meta deseada. La planeación es un área bastante activa dentro de la investigación de la IA y se define como el procedimiento o guía para objetener un objetivo o tarea . Es decir, la busqueda de configuraciones para encontrar el camino con las operaciones necesarias que lleven a la meta. Para esto se ayuda de primitivas operacionales que conforman un patrón de comportamiento. En el caso del sistema del agente robot se tiene como primitivas operacionales las dependencias conceptuales, y en el sistema emocional ordenes en las cuales se muestra la emoción y si esta se debe aumentar o disminuir de acuerdo al contexto que se posea. Generar un plan que lleve a una meta implica un análisis de los pasos intermedios necesarios. El ejemplo presentado por el sistema del robot es un buen ejemplo de planeación.

La temporización se encarga de decidir que primitivas operacionales se ejecutarán en el siguiente ciclo de control. Esta fase no se incluyó dentro del diseño del sistema debido a que el sistema experto la contempla dentro de su estructura (Función que realiza la agenda, otorgando prioridades a las situaciones).

En la ejecución, lo que se obtuvo en ambos sistemas es la ejecución de primitivas que accionan las funciones básicas de movimiento, comunicación o sensoriales. Por último la capa de cooperación con otros agentes incluye la ejecución de protocolos de negociación y creación de compromisos conjuntos entre los agentes. Ninguno de los dos sistemas se desarrollaron con modulos cooperativos aunque se tiene la capacidad de adición.

En conclusión este modelo conceptual de agentes se muestra como un modelo por capas, que definen el estado interno de los agentes (percepción, creencias, situaciones, metas, primitivas operacionales e intenciones) asi como sus funciones básicas que modifican los estados mentales (reconocimiento de situaciones, activación de metas, planeación, temporización y ejecución) y que explican las relaciones funcionales entre lo que un agente percibe y lo que hace.

Capítulo 4

Procesamiento del Lenguaje Natural

Lenguaje Natural es el lenguaje que usamos para comunicarnos con otras personas y que de igual manera se puede usar como interface entre humanos y máquinas. Al usar lenguaje natural un usuario debe de poder hacer preguntas y dar comandos en una manera natural. Si estos comandos son dirigidos a una computadora, esta tiene que entenderlos para responder adecuadamente [SAVA95]. Existen una serie de definiciones sobre el lenguaje natural que a continuación se muestran.

La definición operacional de comprensión es cuando un sistema realiza las acciones que el usuario le pide. Esta definición asume que las acciones que la computadora toma son las correctas.

La comprensión del lenguaje natural (CLN) se refiere al entendimiento de la computadora hacia el lenguaje humano, y ésto incluye la comunicación tanto hablada como escrita. La mayoría de las técnicas desarrolladas en los ultimos 25 años para la CLN conciernen en su mayoría al análisis sintáctico de sentencias correctamente escritas.

La CLN juega dos roles importantes al traducir enunciados de voz en comandos utilizables por la computadora. El primero es obtener el significado correcto de la voz para que la computadora obtenga el mensaje correcto. El segundo que es el más importante es el reducir la ambigüedad acústico fonética que se encuentra en la voz normal, es decir, entender el significado real. Sin técnicas de CLN los sistemas de reconocimiento automático de voz que usan un análisis púramente acústico-fonético sólo funcionan en palabras pronunciadas lentamente. Esta no es la manera en que hablamos naturalmente y a esto se une el hecho de que al pronunciar las palabras en forma aislada resulta una comunicación tediosa comparada con voz continua. Sin embargo los sistemas de reconocimiento de múltiples palabras son bastante útiles

como el caso de *Dragon Dictate* que incluye el sistema más grande de vocabulario comercial disponible en el mundo [WHIT90].

Como puede verse la comprensión de las computadoras siempre ha traído algunas controversias filosóficas, las cuales se tratan de evitar al definir comprensión como la habilidad de responder apropiadamente a directivas dentro del lenguaje humano basándose en información que ya se posee. La CLN de las máquinas es la decodificación de mensajes codificados con los símbolos y convenciones que los humanos usan entre ellos mismos. Para esto se requiere un uso extenso del contexto pues la comunicación humana es extensamente condicionada por los receptores y transmisores de mensajes. Si la comunicación es escrita los símbolos se restringen a texto y unos pocos símbolos de puntuación y separación. La voz por otro lado contiene más portadores de símbolos, tales como tono volumen y duración del segmento los cuales juntos se dicen suplir la información prosódica. Para entender las convenciones de asignar significados a ciertas secuencias, necesitamos entender los roles de la sintaxis pragmática y semántica, puntos clave en la comunicación humana.

Comunicación es el intercambio de mensajes que describen el estado de un modelo o un cambio en el estado del modelo. Estos modelos son típicamente mentales y pueden ser de igual forma computarizados. Los mensajes que son codificados en forma serial sirven para comprimir información y distribuirla en varios conjuntos de símbolos. La codificación serial y las posibilidades para la información contextual creada son metodologías poderosas de información contextual. Las cuales son fundamentales no sólo para una comunicación eficiente sino probablemente para la inteligencia misma. La información contextual es el ambiente local que proporcionan símbolos cercanos que literalmente redefinen otro símbolo. El hecho que un símbolo dado puede tomar diferentes significados dependiendo del contexto implica que la comunicación puede proceder con un número pequeño de símbolos que podrían ser necesitados de otra manera.

Para apreciar el poder de la información contextual para transformar el significado en secuencias simples, considere dos formas extremas que proveen la compresión más poderosa: la información codificada en cromosomas y fractales. Los cromosomas en las células germinales no especifican directamente cuantas células de algun tipo aparecerán en un adulto. En vez de eso especifican como es que cada una reacciona a contextos diferentes que proporcionan otras células a través de intercambios químicos y eléctricos según el cuerpo humano madura. El comportamiento colectivo de las células cambia según van desarrollandose otras células nuevas. El mensaje completo codificado en un cromosoma no puede ser leído directamente pero debe

permitir la creación de contextos intermedios de manera que permite la expresión de la forma final.

Regresando al problema de la sintaxis pragmática y semántica parecería que direccionan en diferentes niveles de la información contextual codificada y abastecida por las cadenas de palabras. Las reglas de sintaxis conciernen típicamente sólo frases y sentencias en lugar de información difundida sobre párrafos o libros. Los modelos pragmáticos y semánticos lidian con contextos más grandes que aquellos encontrados en sentencias aisladas. Ellos generalmente conforman codificaciones que contienen varias sentencias. Los modelos pragmáticos reflejan estados inmediatos de la mente que pueden encontrarse en el progreso de discursos y exposiciones. Los modelos semánticos imparten significado a los mensajes y son los objetos cambiantes o en cierto estado que describen los mensajes. También son llamados mensajes mundiales y están en las últimas capas de un sistema de inteligencia [SAVA95].

Esta es la verdadera frontera entre la CLN e IA. La noción detrás de los modelos semánticos de CLN es que el significado puede derivarse de modelos en el dominio del discurso en lugar de una enumeración exhaustiva de secuencias de palabras. Esto reduce el problema de reconocer cadenas de palabras de un posible número de manipulaciones finitas que pueden ser ejecutadas en un modelo finito. En otras palabras pueden existir un número infinito de formas para expresar la misma idea. La manera de manejar esta situación es codificar la información básica una vez y usar procedimientos para generar las diferentes formas de decirlo. Más aún, en muchos dominios del discurso la infomación básica puede ser pequeña y correctamente representada en un modelo semántico con un número relativamente limitado de hechos y propiedades. En este caso el problema de modelar el discurso humano es el encontrar los modelos semánticos correctos y combinarlos con las reglas de transformación lingüística adecuada.

Terry Winograd propone que la mayoría de los intentos de modelar la comprensión del lenguaje en la computadora han seguido la estrategia de lidiar con un solo componente del lenguaje, es decir, separarlo en sus distintos componentes [WINO72]. Estos modelos están constituídos principalmente como un programa sintáctico(Kuno 1965), un modelo de conexiones semánticas (Schank 1971) o un intento de modelar las estructuras de memoria (Quillian 1967). Uno de los problemas a los que se enfrentan estos modelos principalmente es el de traducir adecuadamente el significado de las estructuras sintácticas y las palabras, tratan casi exclusivamente con la comprensión de una sola oración siendo que la comunicación humana no se da en un contexto tan limitado. Siempre nos encontramos comunicándonos bajo un con-

texto el cual ayuda a interpretar la situación siguiente. Muchas de las estructuras del lenguaje dependen de encontrarse dentro de un proceso de comunicación que incluya un parlante y un escucha inteligentes dentro de un determinado ambiente. Este ambiente no solo incluye una situación física y un tópico de discusión, sino del conocimiento que cada participante tiene acerca del mundo y de las otras ideas. Por lo mismo se deriva la importancia de que el programa entienda el lenguaje del sujeto que habla bajo un contexto de la discusión y bajo un dominio limitado del modelo.

En una plática el individuo toma ventaja de una variedad de mecanismos que dependen de la existencia de un escucha inteligente que usará todo tipo de conocimientos para obtener la información necesaria. Al tratar de hacer un modelo del lenguaje computacional esto representa un problema serio. Por otro lado es imposible separar un aspecto del lenguaje de los otros o separar el conocimiento lingüístico de una persona del conocimiento de la otra.

4.1 Clasificación de Palabras

El proceso de comprensión del lenguaje natural se puede descomponer en ciertos pasos:

- Señal de Entrada.- Puede ser voz o texto de un teclado. Esta señal es transformada en unidades básicas (palabras) para los próximos pasos.
- Análisis sintáctico. En este paso las palabras se prueban para agruparlas de acuerdo a reglas gramaticales. Y se asegura que formen enunciados gramaticalmente correctos
- Análisis semántico.- En este paso el significado de cada palabra y enunciado es asignado. Esta es la parte más complicada de los tres pasos y a menos que se este manejando un dominio del problema sencillo se requiere de una amplia base de datos sobre el tema a discutir.

Una forma de representar el significado contenido en un enunciado es a través de relaciones de objetos. Durante este proceso se localiza el evento principal y los actores. Se determina los roles que juegan en el evento y bajo que condiciones se realizaron. Algunas veces al encontrar el verbo principal, se pueden asociar los roles posibles y condiciones en que ocurre la acción.

Roger Schank sugirió que el mundo normal de cada día podría ser reducido a unos cuantos cientos de modelos genéricos con 30 acciones que podrían realizar

[SCHA81]. Enunciados analizados fueron interpretados como la paráfrasis de estas acciones básicas y la finalidad fue ser guiados por redes de dependencias conceptuales. Noam Chomsky, famoso por trabajar en el área de gramática, transformó un conjunto finito de estructuras de enunciados básicas en un número infinito de maneras de representarlas. La semántica es codificada a un nivel estructural de cierta profundidad. Transformar la gramática juega un rol de dependencia conceptual al traducir información contenida en las estructuras profundas en una forma expresiva que se encuentra en la comunicacion humana. En suma el proposito de CLN es crear modelos semánticos en los cuales se generen significados para después desarrollar codificadores/decodificadores pragmáticos y sintácticos y generalizar el significado de una serie de palabras. La voz continua se distingue por una gran variedad de reglas que sigue y que a la vez viola. Se espera que los enunciados escritos sigan la mayoría de las reglas la mayor parte del tiempo. Sin embargo cualquier discurso espontáneo tiene al menos algunos errores probabilísticos la mayor parte del tiempo.

En su libro de texto, Colim Cherry observa que existen ciertos elementos básicos dentro del discurso que contienen la información base como son palabras clave para comunicarse a pesar de los errores gramaticales, por ejemplo: mujer, calle, tráfico, ruido, robo, bolsa, perder, gritar... lo cual debe implicar que aunque el parlante no emite un discurso perfecto esto no debe detener el proceso de CLN, sólamente hacerlo mas lento.

4.2 Dependencias Conceptuales

En esta sección se muestra como transformar un enunciado de lenguaje natural a una estructura que pueda ser manipulada fácilmente. Estas estructuras ayudan durante el proceso de inferencia, aunque no por eso resuelven totalmente el problema. Lo que se obtiene es una reducción del número de verbos a un menor número de actos, de los cuales se puedan realizar inferencias.

El principal problema que presenta la comprensión del lenguaje es la representación del significado en un lenguaje libre. Básicamente la comprensión del lenguaje puede expresarse como una base conceptual cuyas palabras son mapeadas durante la comprensión. Más aún se puede asumir la base conceptual como la predicción del tipo de información conceptual con la que se puede seguir la entrada inicial. Dependencia Conceptual (DC) es una teoría desarrollada por Schank en los 70's para representar el significado contenido en sentencias. Esta técnica encuentra la

estructura y el significado de una sentencia en un solo paso. Es una representación útil cuando no existe una gramática estricta asociada a las sentencias, y cuando se desea tener inferencias de éstas. Una de las ventajas de la DC es que permite a la computadora hacer inferencias de un sistema de lenguaje natural en la misma manera que los humanos lo hacen.

La representación en DC de una sentencia se construye utilizando primitivas conceptuales y no con primitivas de las palabras contenidas en la sentencia. Estas primitivas representan pensamientos y relaciones entre estos mismos. El uso de DC facilita el uso de reglas de inferencia porque la representación por sí misma contiene estas inferencias. Existen varias primitivas para representar acciones, de las cuales las más comunmente usadas son [SCHA81]:

ATRANS Transferencia de una relación abstracta (ej: dar)
PTRANS Transferencia lugar físico de un objeto (ej: mover)
PROPEL Aplicar una fuerza física a un objeto (ej: empujar)
MOVE Movimiento voluntario de una parte del cuerpo (ej: patear)
GRASP Tomar un objeto (ej: agarrar)
INGEST Ingestión de un objeto (ej: comer)
EXPEL Expulsión un objeto del cuerpo humano (ej: respirar)
MTRANS Transferencia de información mental (ej: decir)
MBUILD Construir nueva información a partir de vieja (ej: decidir)
SPEAK Producción de sonidos (ej: hablar)
ATTEND Enforcar los sentidos en un cierto estímulo (ej: escuchar)

Cada primitiva representa varios verbos que tienen significado similar. Por ejemplo dar, comprar, robar y tomar tienen el mismo sentido final que es la transferencia de un objeto de una entidad a otra. Las primitivas se representan por un conjunto de reglas y estructuras de datos [SCHA81]. De igual manera cada primitiva contiene componentes básicos:

- **Un Actor:** que es la persona que realiza el Acto.
- **Un Acto:** realizado por el actor, y ejecutado sobre un objeto.
- **Un Objeto:** en el cual se ejecuta la acción.
- **Una Dirección:** el lugar al cual tiende a dirigirse el Acto.
- **Un Estado:** que es el estado físico en el cual se encuentra el objeto, en este caso se representa con una escala de valores númericos. Por ejemplo. La salud

puede tener valores que van de -10 a 10, teniendo como un estado perfecto 10, 0 como valor normal y -10 como muerte.

A continuación se presentan algunos ejemplos de DCs y su representación [SAVA95]:

PTRANS, se dijo es la transferencia de lugar físico de un objeto y se representa como:
*(***PTRANS** *(ACTOR NIL)(OBJECT NIL)(FROM NIL)(TO NIL))*
Por ejemplo, el enunciado: *Robot ve a la cocina*, asumiendo que el robot esta en la sala posee la siguiente representación:

*(***PTRANS** *(ACTOR Robot)(OBJECT Robot)(FROM sala)(TO cocina))*

Cuando el verbo ir (ve) se reconoce en el enunciado se habilita una estructura ptrans. Los campos vacios (NIL) se llenan con los elementos que el enunciado facilite. El actor viene a ser el robot y como objeto el robot, tomando encuenta que el robot se mueve a sí mismo de la sala a la cocina.

MTRANS es la transferencia de información mental:
*(***MTRANS** *(ACTOR NIL)(OBJECT NIL)(FROM NIL)(TO NIL))*
Esta DC puede representar el enunciado: *Roberto le dijo a Susan sobre eso*, esta representado por:

*(***MTRANS***(ACTOR Roberto)(OBJECT eso)(FROM cerebro_de_Roberto)*
(TO cerebro_de_Susan))

ATRANS es la transferencia de una relación abstracta u objeto, y su estructura es:
*(***ATRANS** *(ACTOR NIL)(OBJECT NIL)(FROM NIL)(TO NIL))*
Representando el enunciado *Juan da el libro a Maria* queda:

*(***ATRANS** *)ACTOR Juan)(OBJECT libro)(FROM Juan)(TO Maria))*

INGEST es la ingestión de un objeto y cuenta con la representación:

*(***INGEST***(ACTOR NIL)(OBJECT NIL)(FROM NIL)(TO NIL))*
Por ejemplo la frase: *Juan bebe leche* se presenta como:

*(***INGEST***(ACTOR Juan)(OBJECT leche)(FROM nil)(TO boca_de_Juan))*

Aquí no sabemos donde se encuentra la leche, para completar esta estructura se necesita información, ya sea previa, que permita inferir, buscarla en el siguiente enunciado o a través del contexto que se utiliza.

PROPEL es la aplicación de una fuerza física a un objeto y con la siguiente representación:
*(***PROPEL***(ACTOR NIL)(OBJECT NIL)(FROM NIL)(TO NIL))*
El enunciado *Maria mato un insecto al tirarle un zapato* se muestra como:

*(***PROPEL***(ACTOR Maria)(OBJECT zapato)(FROM Maria)(TO insecto))*

En este momento el estado físico del insecto cambia de un valor mayor de -10 a +10. El actor necesita mover su brazo para lanzar el zapato, lo cual significa que se infiere que la acción MOVE se ejecutó. Por lo que MOVE es una primitiva utilizada por **PROPEL**. Como en este ejemplo, una acción puede requerir de otras primitivas para realizarse.

Las preguntas se pueden manejar al utilizar la letra Q en el slot del cual se desea preguntar. Por ejemplo *dónde esta el perro?* puede representarse como:

*(***PTRANS** *(ACTOR perro)(OBJECT perro)(TO Q)*
(FROM lugar_inicial_perro))
y el sistema busca el lugar respondiendo con una primitiva MTRANS que es:

*(***MTRANS** *(ACTOR computadora)(OBJECT "lugar_nuevo_perro")*
(TO persona)(FROM computadora))

Como se dijo las DC's se utlizan para representar acciones simples. Y de igual forma sirven para representar comandos o preguntas simples, pero no son muy utiles al tratar de representar enunciados más complejos. Schank y Owens explican que para simbolizar el siguiente enunciado tomaría alrededor de 2 páginas de DC's: "Juan aposto a Sam 500 pesos que los toros del Neza ganarian el juego final del fútbol".

Finalmente la teoría de DC es una teoría de representación de eventos. Pero para representar la manera en que una máquina entiende es necesario tener otros elemen-

tos de ayuda extra. Sin embargo las DC tienen la ventaja de ser independientes del lenguaje, facilitar la capacidad de inferencia y proporcionar poder predictivo al proceso de entendimiento. Estas caracteristicas las convierten en una buena representación para traducciones del lenguaje, comprensión de historias, o cualquiera aplicación que requiere comprensión y manipulación de los conceptos que muestran los enunciados. [SIMM86] La técnica de Dependencia Conceptual explicada en esta sección se incorporó en el sistema experto desarrollado en el presente trabajo.

4.3 Guiones

La idea de que un objeto constituye o forma parte de otro objeto es bastante conocida. De igual forma una descripción genérica de una acción constituída de otros eventos sencillos es llamado guión o *script*, particularmente en el área de la Comprensión del Lenguaje Natural. Algunas aplicaciones de este tipo se basan principalmente en la habilidad de reconocer instancias de eventos esquemáticos que muestran descripciones de sus partes. De esta manera el texto se puede organizar en eventos coherentes [RUSS95]

Una manera de representar el contexto es a través de guiones. El término guión es tomado del lenguaje utilizado en la actuación y describe posibles secuencias de eventos que un actor o entidad puede realizar bajo ciertas condiciones. Al utilizar los guiones es fácil obtener inferencias acerca de los eventos sobre los que se tiene información incompleta o difusa. Los guiones organizan el conocimiento y lo decodifican. Cada guión tiene diferentes roles asociados, los cuales se completan con actores y objetos en el momento de crearlo [SAV95]. Nosotros usamos nuestro conocimiento en situaciones diarias, el cual nos ayuda a entender historias acerca de esas situaciones. Por ejemplo para entrar al cine no necesitamos preguntar porque alguien tiene que checar nuestro boleto, ni porque es necesario mantener silencio durante la película.

Para comprender una historia o sucesión de eventos basada en guiones, primero hay que identificar el guión al cual se está refiriendo. A este reconocimiento se le llama "ocurrencia de guiónes". A continuación el guión es utilizado para llenar los detalles importantes en la secuencia de eventos que está sucediendo. En este momento se crea una aplicación de guiones. Esta aplicación llena los eventos que no se mencionan dentro de una secuencia partiendo de dos eventos similares no relacionados. Estos dos procesos de comprensión poseen sus contrapartes. Por ejemplo cuando alguien relata una historia, lo más seguro es que no menciona todos

los eventos que ocurren para evitar el aburrimiento de los demás, asume que el escucha tiene cierta familiaridad con el guión y entenderá según ciertos elementos cruciales se mencionen.

Un guión de una cafetería contiene una gran cantidad de información que se conjunta con una amplia variabilidad de eventos que ocurren dentro de la misma. Y dentro de la cafetería se tienen escenas de entrar, ordenar y pagar pero con un conjunto de posibilidades diferentes a las de un restorante elegante por ejemplo. Cada guión posee un número de papeles asociados, cuando este guión se identifica los actores en la historia asumen los papeles dentro del guión. Si un actor no ha sido específicamente mencionado se puede asumir su presencia. La naturaleza de los detalles inferidos en un guión depende del número de eventos que éste presente. Considere la siguiente historia:

Juan fué a una cafetería. Ordenó pollo y dejó una buena propina.

Normalmente el guión se divide en varias etapas denominadas escenas. La acción de ordenar crea la escena de ordenar en el guión de cafetería. Los eventos que ocurren entre ordenar y dejar propina, así como las escenas de salida también se asume que suceden. En sí la historia se debería de entender como si fuera:

Juan entró a la cafetería
Se sentó
Leyó el menu
Ordenó pollo
Comió el pollo
Dejó una buena propina
Pagó la cuenta
Dejó la cafetería

Como se ve este ejemplo presenta una serie de pasos como si se hubiera escuchado toda la despcripción de la escena. Pues cada paso precede al anterior. Pero alguien que trata de entender esta escena debe de poseer el conocimiento necesario para saber que pasos se dieron y en que orden se mostraron. Así como también saber cuales son los pasos siguientes a ejecutar.

4.3.1 STRIPS

Fikes, Hart y Nilsson [HUTC95] proponen una manera de desarrollar esto, ellos hablan de un sistema donde se puedan crear nuevas acciones llamadas STRIPS. Cada nuevo operador que aparece es una secuencia de otros operadores viejos y más

sencillos. Esta teoria se aplica en el reconocimiento de operaciones similares. Un ejemplo se observa al darle a un robot una tarea tal como "trae la pelota al cuarto del niño" y lleva la pelota de la sala al cuarto sugerido. STRIPS almacena estos planes generalizados de manera que cualquier parte del plan se pueda aislar y ejecutar de manera separada. En caso de que el plan se pierda a la mitad de su ejecución, el sistema tendrá la información necesaria para analizar la razón. STRIPS tiene un conjunto de reglas que producen nuevos modelos de modelos antigüos donde cada regla consiste de:

- un nombre con parámetros
- una condición
- una lista de elementos a borrar
- una lista de elementos a añadir

Una instancia de STRIPS se forma al darle valores a estos parámetros. Al accionar las reglas se crea un nuevo modelo constituido de:

- todos los hechos en el modelo original
- menos todos los hechos en la lista de elementos a borrar
- más todos los hechos de la lista de adición

El capítulo 7 se mostrará una aplicación interesante que combina el uso de Dependencias Conceptuales y STRIPS para reconocimiento de situaciones en un entorno computacional.

Capítulo 5

Emociones y su Representación

El sistema de control para la interfaz humana para comunicar emociones interactúa con interfaces de despliegues faciales, este capítulo trata de mostrar las bases psicológicas y los trabajos realizados, así como dar una breve introducción a las aplicaciones que se pueden desarrollar con estos despliegues.

5.1 Despliegues Faciales y Discretización de las Emociones

Los despliegues faciales han sido objeto de estudio de muchos científicos por un largo tiempo. Su estudio ha atraído el interés de un diferente número de disciplinas, incluídas estan la psicología, etología y todas aquellas relacionadas con comunicaciones interpersonales. Históricamente Darwin fue el primero en identificar dos aspectos de las expresiones faciales en su famosa publicación *La expresión de emociones en el hombre y animales* en 1882; el primer aspecto esta relacionado a la emoción por sí misma y la segunda se refiere a la comunicación. Mas aún, el campo de expresiones emocionales ha tenido contribuciones invaluables para el análisis y entendimiento de las expresiones faciales. Fridlung and Gilbert propusieron que el rol primario de un despliegue facial consiste de proporcionar información que aumente el componente verbal de la comunicación en lugar de proveer información emocional.

El mayor acercamiento de estos estudios se obtuvo cuando Woodworth (1938) propuso el primer sistema realmente exitoso de clasificación de las expresiones faciales (emociones discretas). Desde entonces han existido una gran cantidad de intentos por categorizar estos despliegues de acuerdo a sus roles comunicativos.

Uno de los más conocidos es el propuesto por Eckman and Friesen [ECKM75], que crearon una categorización universal, que incluye seis emociones básicas que un humano puede desplegar: sorpresa, miedo, desagrado, enojo, felicidad y tristeza. Ellos mencionan que los despliegues faciales son los mismos en todo lugar y tiempo que aparezcan. Y la categorización de emociones hace que los despliegues sean dependientes de la situación al momento de comunicarse.

A este respecto se puede asumir que emociones intensas tales como enojo, pasión, odio, miedo y avaricia son experimentadas universalmente como es propuesto también por Eckman e Izard [ECKM75] [IZAR77]. Las expresiones faciales espontáneas de emoción son reguladas por factores sociales que los individuos manejan de manera adecuada en la presencia de otros. A la fecha se han reportado un gran número de investigaciones en esta área de la interpretación facial, con experimentos que incluyen sujetos de varias edades y razas.

Izard en su trabajo presentado [IZAR77] intenta encontrar una definición completa de emoción y señala que debe incluir tres componentes:
(a)la experiencia o sentimiento conciente de la emoción,
(b)el proceso fisiológico que ocurre en el cerebro y sistema nervioso y
(c)la presencia de patrones expresivos para la emoción, particularmente aquellos mostrados en la cara.

De igual manera Izard clasifica en diez categorias las emociones fundamentales y son: interés o excitación, felicidad, sorpresa, angustia, enojo o rabia, desagrado o repulsión, indiferencia, miedo o terror, timidez o humillación, y culpa. Y aunque estos son los básicos existen factores que pueden afectar el despliegue emocional, como por ejemplo el amar u odiar.

Es importante mencionar que Izard también posee estudios sobre despliegues faciales en infantes y trata de explicar el desarrollo de los despliegues emocionales a través de la vida de una persona. Así mismo afirma que ciertas expresiones faciales de la emoción son innatas, universales y homólogas a las expresiones encontradas en primates. Algunas de estas expresiones (incluyendo toda la cara y configuraciones completas) se encuentran presentes en el nacimiento mientras que otras como es la indiferencia, emergen en los primeros meses de vida [IZAR92].

La investigación realizada por Nagao y Takeuchi afirma que las expresiones faciales son también vistas de dos maneras. La primera se refiere a expresiones faciales como comunicación de estados emocionales. Y la otra ve las expresiones faciales en un contexto social y hace referencia ellas como señales de comunicación, que pueden ser redundantes o no. Aunque el término "despliegues faciales" es equivalente al de

"expresiones faciales" el segundo no posee la idea de comunicación. Chovil assegura que una función primaria de un despliegue facial es el de comunicar mensajes a otros [CHOV92]. Sus estudios muestran los despliegues faciales como elementos lingüísticos de un mensaje. Su propósito principal es el descubrir la manera en que estos despliegues contribuyen en la conversacion, y los clasifica en categorías de acuerdo a la función realizada, es decir si es redundante o no con el contexto verbal.

Otra de las razones para poner atención en la diferentes facetas de una cara es que estas son una interface social. Como lo menciona Takeuchi las futuras tecnologías deberán ayudar a la gente a establecer y reforzar sus relaciones sociales. Los humanos son seres sociales y los despliegues faciales son por lo general, dirigidos no a uno mismo sino a los que nos rodean. Estos han evolucionado de manera que nos ayudan a desarrollar mejores relaciones sociales con los demás.

Finalmente para poner en claro la importancia de los despliegues faciales muchos experimentos han demostrado que éstos despliegues son de gran ayuda especialmente para un primer contacto con el sistema. Se ha mostrado también que una interacción temprana con estos despliegues faciales aumenta la probabilidad de una exitosa comunicación, aun cuando no existen los despliegues faciales en la interacción futura, solo en el principio. Por lo cual se puede decir que las interfaces con despliegues faciales reducen la barrera mental entre los usuarios y los sistemas de cómputo. En el caso del presente trabajo se buscó un punto de vista en el cual el despliegue de las emociones sirvan a esta labor mencionada y se mantenga un sistema de despliegue continuo y actualizado.

5.2 Reconocimiento de Palabras Clave, Emociones y su Cuantificación

La mayoría de los sistemas que emplean reconocimiento del lenguaje natural se han desarrollado para propósitos especiales como son interfaces de lenguaje natural para información de bases de datos (por ejemplo información telefónica de los horarios de llegadas y salidas en los trenes). Los sistemas sofisticados de procesamiento del lenguaje natural tratan de *entender* el lenguaje al extraer conocimiento de la estructura gramatical con que se construyen las sentencias. Y también se ayudan a través de características especiales para crear inferencias y explicar el razonamiento. Por otro lado sistemas de diálogo como ELIZA (un sistema computacional que simula una persona con la que se puede conversar) y sus descendientes fueron desarrollados

tomando un reconocimiento de palabras clave (también llamados sistemas *pattern matching*). Estos sistemas de diálogo usan reconocimiento de palabras y utilizan patrones de reconocimiento que conducen los diálogos hacia un dominio especial, el cual se mantiene sin cambio durante la conversación.

ELIZA actua en el rol de un psicólogo terapista y no tiene habilidades de entendimiento del lenguaje, simplemente lee un enunciado de entrada, aplica sus patrones de reconocimiento y genera una respuesta. Por ejemplo en el enunciado de entrada: *Mi X es Y* puede producir una respuesta de la forma *>Porqué dices que tu X es Y?*. Cada patrón de entrada satisfecho posee diferentes respuestas posibles asociadas a él. Sin embargo, aunque ELIZA aparece como un sistema bastante convincente no es dificil llegar a encontrar sus errores al tratar de hacerlo comprender conceptos [GERM92].

En el presente trabajo, la parte que se encarga del reconocimiento del texto incorpora un reconocimiento de palabras que introduce el usuario y que concuerdan con palabras claves que el sistema desea aislar. Una vez alcanzada esta operación con éxito se procede a realizar una hipótesis emocional acerca del despliegue que el usuario trata de comunicar.

Para hacer más clara la explicación comencemos con un ejemplo simple donde dos personas hablan entre sí tratando de expresar sus sentimientos hacia la otra:

Juan: *Estoy muy enojado contigo*

Jim: *Bueno, yo me encuentro resentido y temeroso*

Dyer fue uno de los primeros que extrajo palabras clave de los enunciados anteriores como es el caso de la palabra *enojado* para caracterizar el estado emocional de Juan. De igual forma el estado emocional de Jim se caracteriza con la palabras *resentido* y *temeroso*. Estos estados son almacenados en los objetos de implementación correspondientes a las emociones, rastreando de esta manera la emoción anterior que se mostró.

Es importante ahora, tomar en cuenta que un gran número de estados emocionales pueden ocurrir dentro de una misma conversación. Además de que diferentes estímulos perturban la intensidad de estos en diferente grado, y este efecto puede ser con mayor frecuencia para estados emocionales extremos. Por ejemplo un golpe físico perturbará el estado de una persona de neutral (digamos grado=1, en una escala 1-4) al estado enojado (grado=3), pero puede suceder que la persona ya se encontraba irritada (grado=2) y se perturba de tal manera que llega a sentir rabia (grado=4). De igual manera un elogio puede modificar en forma opuesta estos estados por ejemplo de enojado a neutral.

Existen razones importantes para tratar de cuantificar las diferentes emociones en una escala de medición y otorgarles un valor número. La principal es debida a que usamos computadoras, cuyo funcionamiento se basa en el intercambio de datos numéricos, para representar estas emociones dentro de un mundo virtual. Por otro lado, el problema de las mediciones no es tan trivial pues bien se sabe que todo efecto en la naturaleza no es algo medible en forma discreta. Por lo que para poder medir emociones se deben tener ciertas consideraciones: primera que sean cambios analógicos y segunda, tomar en cuenta que estos cambios pueden ser tanto muy notables como imperceptibles.

Los trabajos presentados por Schamlz y Dankel [SCHM95] así como el de Piesk y Trogemann [PIES97] proponen que los cambios sobre una misma emoción pueden modelarse como una función exponencial. Piesk y Trogeman crearon un actor virtual que actua siguiendo guiones dentro de una historia, que el actor se encarga de narrar. Ellos proponen que todos los estados emocionales pueden ser representados por una combinación de emociones básicas y utilizan las identificadas por Izard. Esto se hace otorgando valores reales a los diferentes niveles de un estado emocional E que se comporta como una sumatoria de pesos de todas las emociones experimentadas. La emoción más reciente es la que más contribuye a este estado, claro esta. Para cada emoción básica j existe un valor número que es computarizado en forma separada usando la siguiente expresión:

$$g(j) = e^{-i} \tag{5.1}$$

donde:
j= Cada emoción básica
g= valor de intensidad
i= marca el cambio de intensidad de la emoción

Cada valor especificado es un factor que contribuye al cálculo de la emoción actual del actor virtual. La intensidad actual em de la emoción presente es modificada en tiempo real al ir sumando todos los impulsos emocionales imp_j que corresponden a los valores números de las emociones previamente activadas. Conjuntando todo, el valor actual del estado emocional actual se calcula con la ecuación:

$$em = \sum_{j=0}^{N} imp_j \times e^{-i} \qquad \forall j > 0 \tag{5.2}$$

donde:
em=intensidad de la emoción presente
N=Número constante de impulsos emocionales que modifican la emoción presente
j=es el índice de cada emoción participante
imp_j=impulso emocional j que contribuye a obtener la emoción presente
i=marca el valor de cambio en las emociones.

Schmalz hace un proceso similar, sólo que el incluye factores ambientales que afectan y hacen el cálculo matemático más complicado. Pero de igual forma ellos proponen un comportamiento exponencial para cada estado emocional que se presenta [SCHM95] [SCHM95b].

5.3 Modelos de Emociones

5.3.1 Clasificación de las Emociones

Las expresiones faciales pueden ser vistas en dos formas. Una haciendo alusión a expresiones que tratan de mostrar emociones. Y la otra ve a las expresiones faciales en un contexto social tratándolas como señales de comunicación. En el presente trabajo se intenta hacer un modelo en el cual se posea una combinación de éstos últimos.

Claramente es necesaria una clasificación de emociones básicas a utilizar. Aunque la literatura encontrada en el área de psicología esta repleta de diferentes taxonomías aunado a que existe muy poca concordancia entre ellas.

Para este trabajo se escogió un sistema de clasificación general similar al propuesto por Izard [IZAR77]; en el cual las expresiones faciales se clasifican en siete estados fundamentales: interés, felicidad, sorpresa, tristeza, enojo, miedo y disgusto. Esto sugiere que el sistema solo necesita clasificar las entradas de texto adecuadamente para cada emoción con una medición de la intensidad en todas las categorías. Y una vez que ésta intensidad rebasa un cierto umbral es cuando el avatar cambiará su despliegue facial para representar la emoción apropiada.

La Tabla 5.1 resume algunas de las diferentes emociones y la manera en que se afecta la intensidad de las mismas en una manera discreta. Estas emociones varian de acuerdo a perturbaciones dando lugar a una subcategorización dentro de las emociones fundamentales. Es decir, para felicidad un cierto estímulo hará que varíe de neutral (grado=1) a feliz (grado=3), o contento (grado=2) a muy feliz (grado=4). De igual manera, puede suceder en sentido contrario con un estímulo negativo.

Emocion	1	2	3	4
Felicidad	neutral	contento	feliz	muy feliz
Enojo	neutral	molesto	irritado	muy enojado
Tristeza	neutral	melancólico	triste	deprimido
Miedo	neutral	incómodo	temeroso	aterrorizado

Tabla 5.1. Modelo Emocional.

Esta representación contiene un número razonable de posibles estados emocionales y respuesta a las perturbaciones, que hacen que el sistema posea considerable flexibilidad aunado a una base de conocimientos lo suficientemente capaz de crear datos sin crear problemas en el espacio de almacenamiento.

5.4 Logica Difusa en las Emociones

Debido a que se desean cambios continuos en las emociones presentes, es necesario un proceso de cuantificación de manera que se puedan observar estos cambios de una manera más natural. Los cambios deben ser continuos y lo mejor es representarlos en una forma analógica para que los avatars tengan cambios naturales al pasar de un estado digamos enojado a alegre. Estos cambios puede dar la idea de información imprecisa cuando se trata de encontrar el grado en el cual una emoción pasa de un estado a otro.

La lógica difusa es un método de representación sencillo utilizado para procesos analógicos en una computadora digital. Estos procesos incluyen fenómenos que no son fáciles de cuantificar como segmentos discretos, y los conceptos que incluye son dificiles de modelar algunas veces [COEX92]. No solo se otorga un valor de intensidad a una cierta emoción y se decide si cae en un conjunto único de emociones, normalmente existen traslapes sobre estos conjuntos en los cuales es díficil discernir los límites de una frontera a la otra. A continuación se explica con mayor detenimiento el proceso de cuantificación en la lógica difusa.

5.4.1 Lógica Difusa

En esta sección se discuten bases teóricas de lógica difusa que se utilizaron para el sistema emocional. Esta teoría tiene inferencia con la cuantificación y razonamiento

utilizando lenguaje natural en el cual las palabras poseen significados ambiguos como son: poco, medio, bastante y otras relacionadas. También permite el desarrollo de sistemas continuos que representan cambios de la naturaleza en forma analógica. Esta teoría se desarrolló a partir de los conjuntos difusos introducidos por Lofti Zadeh [ZAD88]. Desde entonces se utiliza en un gran número de aplicaciones como son: algoritmos de control, diagnósticos médicos, toma de desiciones, ingeniería, psicología por nombrar algunas. Para iniciar una introducción es necesario introducir la noción de conjuntos difusos.

Conjuntos difusos

Comúnmente para representar un objeto como miembro de un conjunto se toma en cuenta una función característica en la cual si el elemento pertenece a ella la función tiene valor 1 y si no su función tiene valor 0. Es decir, o pertenece o no pertenece completamente.

$$\mu_A(x): \qquad X \rightarrow \{0,1\} \tag{5.3}$$

Esta definición da el concepto clásico del mapeo del conjunto X al conjunto 0 y 1. Nótese que estos conjuntos son mutuamente exclusivos, por lo que nos encontramos con una clase de indeterminación, que no permite miembros parciales. La teoría de conjuntos difusos puede expresar conceptos inexactos como el rango de valores entre 0 y 1 o ambos.

Esta subjetividad tiene implicaciones profundas para modelar sistemas continuos y además es el poder y flexibilidad de la lógica difusa. La lógica difusa permite una transición gradual entre ser completamente miembro de un conjunto y no completamente miembro del conjunto. Y ahora nuestra función se denomina función de pertenencia:

$$\mu_A(x): \qquad X \rightarrow [0,1] \tag{5.4}$$

Ahora la función de pertenencia mapea X en el codominio de los números reales definidos en el intervalo de 0 a 1, es decir, la función resulta en un número real:

$$0 \leq \mu_A \leq 1 \tag{5.5}$$

Un valor particular de la función de pertenencia como puede ser 0.5 se le denomina grado de pertenencia o de verdad. μ es la forma de denominar este grado en el

cual un elemento es miembro de un conjunto difuso, en un rango de 0 a 1.

Los conjuntos difusos son por lo general utilizados en el lenguaje natural y ejemplos de esto pueden ser:

Josefina es *alta*

Si la masa es *demasiado densa* agrege *mucha* agua.

Donde las palabras itálicas se refieren a conjuntos difusos y cuantificadores que se pueden representar y operar en la lógica difusa. En estas proposiciones se pueden observar en el primer ejemplo en el cual Josefina es alta dependiendo de cierto grado: poco, algo, bastante, mucho. Estos calificadores se utilizan para modificar el conjunto difuso. Muchas de las palabras utilizadas en el lenguaje natural se pueden de definir en término de los conjuntos difusos como se verá más adelante.

Retomando el primer ejemplo en el que Josefina es alta. Dentro de la lógica clásica no está bien definido este conjunto, pues la estatura va en una escala de bajo hasta alto y la sensación de mujeres altas se vuelve gradualmente débil para después ser gradualmente fuerte. Por lo que si los grados de verdad (0 y 1) son designados nuevamente para representar esta sensibilidad el resultado es una curva como la mostrada en la figura 5.1

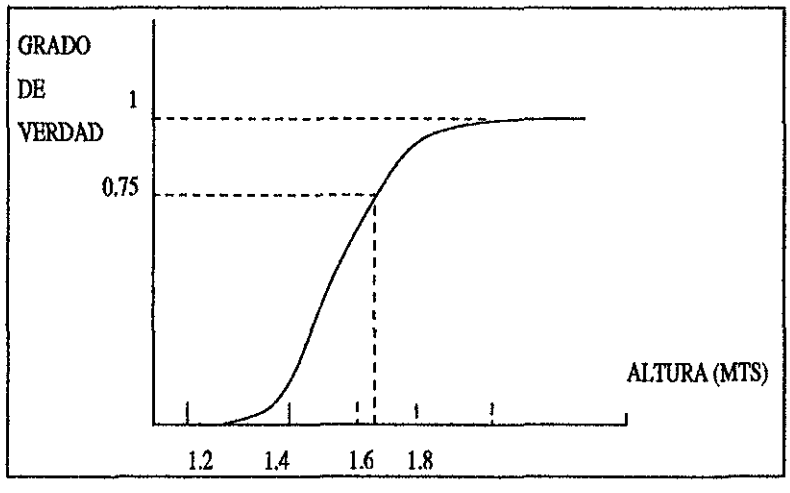

Figura 5.1: **Conjunto difuso de las mujeres altas.**

La lógica difusa permite la afirmación Josefina es alta para tener un rango de veracidad que, en este caso, depende de las demás mujeres altas. Por ejemplo si Josefina mide 1 metro la afirmación de que es alta es completamente falsa; pero si ella mide 2.2 mts. es entonces completamente verdadera. Si consideramos que

ahora mide 1.65 mts. puede ser 75% verdadera. Trabajando con estas premisas el camino entre verdadero y falso puede ser gradual y además implícito, ya que puede presentarse simultáneamente la verdad parcial y la falsedad parcial. El punto de cruce dentro de estas gráficas corresponde a aquel valor cuyo grado de verdad es igual a $\mu = 0.5$, es decir un 50% de las personas dirá que Josefina es alta y un 50% dirá que no lo es. En este caso si Josefina mide 1.65 mts. se dice que $\mu(1.65) = 0.75$.

Existen dos funciones de pertenencia principales que son generalmente utilizadas: la función S y la función Π.

La función S es una función matemática que se utiliza muy comúnmente en los conjuntos difusos y se define como:

$$S(x; A, B, G) = \begin{cases} 0 & \text{para } x \leq A \\ 2\left(\frac{x-A}{G-A}\right)^2 & \text{para } A \leq x \leq B \\ 1\text{-}2\left(\frac{x-G}{G-A}\right)^2 & \text{para } B \leq x \leq G \\ 1 & \text{para } x \geq G \end{cases}$$

Donde:

$$\begin{aligned} A &= \text{valor\_mímimo en x} \\ G &= \text{valor\_máximo en x} \\ B &= \tfrac{A+G}{2} \end{aligned} \tag{5.6}$$

Una gráfica de la función S se muestra en la figura 5.2.

En esta definición observe los parametros A, B y G que se pueden ajustar para obtener el grado de pertenencia deseado, o este ajuste puede ser solo aproximado. Claro esta que en el último caso esta función de pertenencia se define sin alguna referencia a datos tabulares. Dependiendo de la aplicación también se pueden utilizar otras funciones como las triangulares. Observe que el parámetro B es la mitad de la curva en el punto de cruce. Valores grandes de B corresponden a una curva ancha y valores pequeños corresponden a una curva estrecha.

La función Π da una curva como la mostrada en la figura 5.3 y su valor decrece hasta cero en puntos específicados.

Su funcion es:

$$imp_j\ \Pi(x; B, G) = \begin{cases} \text{S(x; G-B, G-B/2, G)} & \text{para } x \leq G \\ \text{1-S(x; G, G+B/2, G+B)} & \text{para } x \geq G \end{cases}$$

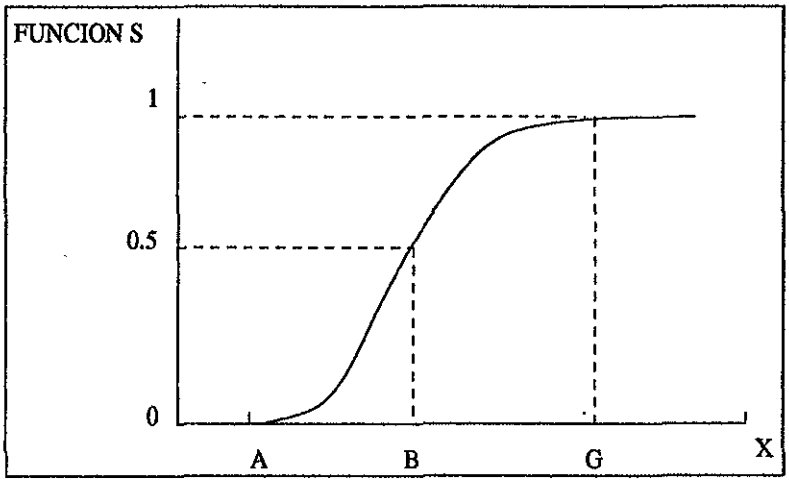

Figura 5.2: **Función S.**

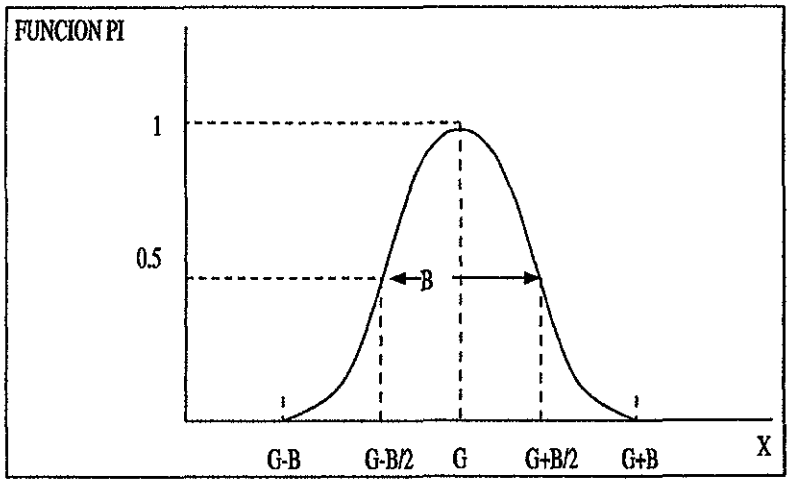

Figura 5.3: **Función π.**

El parámetro B es ahora el ancho de banda o ancho total en los puntos de cruce. La función Π llega a cero en los puntos $x = G \pm B$.

Y los puntos de cruce estan en $x = G \pm B/2$

Variables lingüísticas y su aplicación

Una aplicación importante de la lógica difusa es en la lingüística. La lógica difusa y las variables lingüísticas pueden usarse para cuantificar el lenguaje natural y de esta manera manipularlo. Una variable lingüística es una variable cuyos valores no son números sino palabras o sentencias en un lenguaje natural. Por ejemplo "FELIZ" es una variable lingüística y sus valores pueden ser "MUY FELIZ", "FELIZ" y "POCO FELIZ". Esta variable se puede representar en la siguiente figura que muestra el grado de felicidad dentro del Universo U=[0,100].

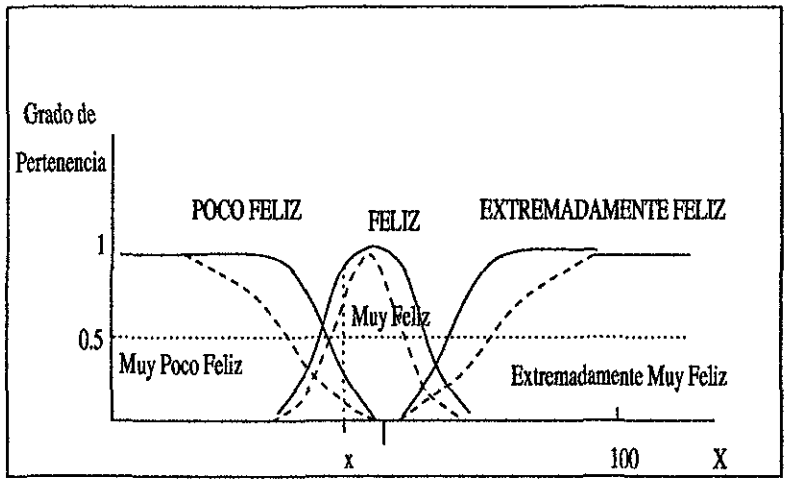

Figura 5.4: **Representación de la variable lingüística FELIZ.**

Observe que las líneas punteadas entre los valores de FELIZ indican que los rangos de estas variables con respecto a la variable base están completamente abiertos para ser especificados según el criterio o razonamiento del diseñador.

El porcentaje de probabilidad de que un valor X de la variable base pertenezca a un valor de felicidad se representa por el número que se incluye en cada una de las líneas. Por ejemplo la probabilidad de que un valor de la variable base como es 45 pertenezca al valor de felicidad "MUY POCO FELIZ" es de .2 pero la probabilidad de que este mismo valor pertenezca al valor de edad "FELIZ" es de .9. Por lo tanto,

sea cual fuere el valor de la variable base; el controlador es capaz de determinar a que valor pertenece basándose en los rangos establecidos para cada valor de felicidad..

En general, los valores de una variable lingüística se pueden generar a partir del término primario (por ejemplo "JOVEN") y una colección de modificadores ("NO", "MUY", "MAS" O "MENOS", "IGUAL", etc). Por ejemplo un valor de "edad" puede ser "no muy joven" y "no muy viejo". Cada valor puede ser generado por un contexto-libre gramátical. Además cada valor de una variable lingüística representa una posibilidad de distribución. La cual se calcula a partir de la probabilidad dada por μ.

Obtención de Salidas

Después de aplicar las variables lingüísticas que modifiquen el valor de la emoción dentro de los conjuntos difusos, se procede a trabajar con los grados de verdad que le corresponden al valor asignado con estos valores μ.

Como se tiene una única variable a utilizar se realiza la suma lógica del resultado obtenido. El primer paso es verificar si este valor modifica a más de un conjunto difuso creando un traslape, tal es el caso de la figura 5.5. Acto seguido se obtiene el valor de μ con el que está afectando a los conjuntos involucrados. En este caso el valor $x1$ afecta al conjunto difuso 2 con un valor $\mu 1$, y al conjunto difuso 3 con un valor $\mu 2$. En este momento se realiza la suma de las áreas de los conjuntos delimitados por $\mu 1$ y $\mu 2$. Esta suma combina el resultado de los conjuntos y conforma un área difusa que corresponde a los valores de entrada.

El grado de pertenencia afectará cierto porcentaje de un conjunto y otro tanto del otro conjunto. Puede suceder que el valor obtenido sólo afecte a un conjunto y entonces únicamente afecta a este conjunto en su totalidad. Ver figura 5.6.

Una vez que se les ha otorgado su valor normal se procede a la finalización del proceso. El resultado obtenido en el proceso anterior se debe transformar en un valor número concréto. Debido a que el área sombreada no puede utilizarse directamente en un control físico de la emoción, se desea transformar dicha área en un valor numérico utilizable para el sistema físico. En este caso se desea un valor número que ejemplifique el grado emocional presente.

Existen varios algoritmos de defuzzificación de salidas que si el lector desea profundizar puede referirse al libro de Cox [COXE94]. Uno de ellos es el algoritmo del centro de gravedad el cual se explica con mayor detenimiento en el capítulo 8. Al

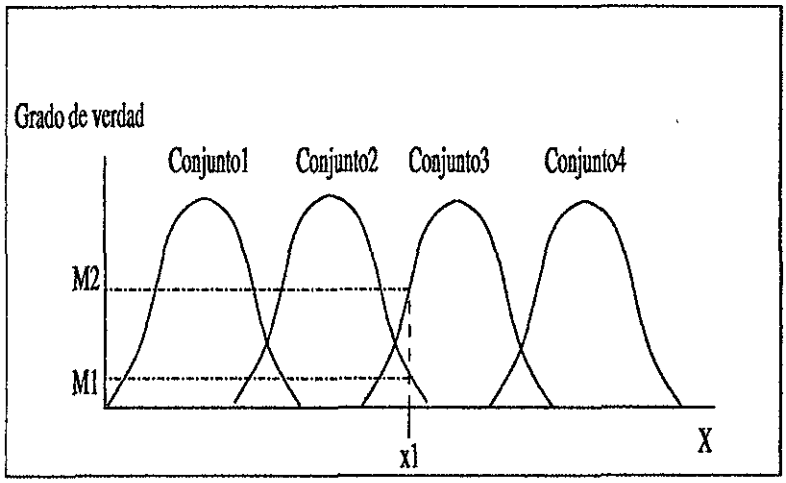

Figura 5.5: **Traslape de conjuntos difusos.**

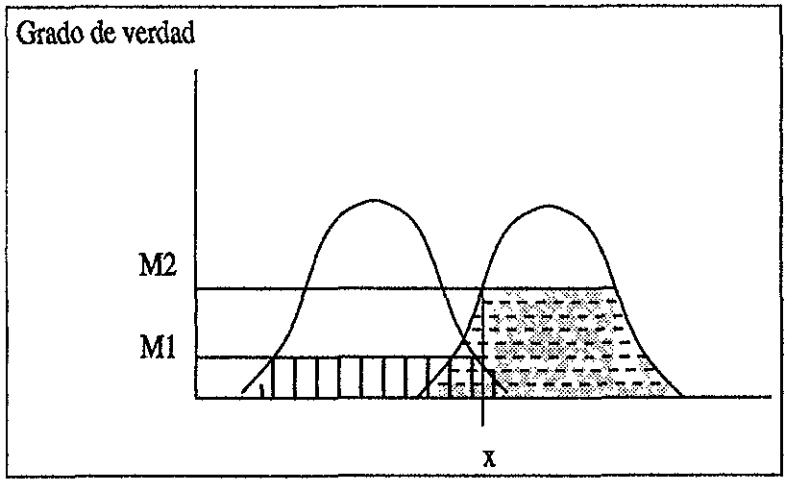

Figura 5.6: **Resultado de suma lógica.**

aplicar este algoritmo se obtiene una salida númerica mas precisa y lista para usarse como resultado de este proceso.

Capítulo 6

Comunicación en Ambientes Gráficos

6.1 Comunicación remota en los ambientes gráficos

Los ambientes gráficos de computadora, denominados Sistemas de Plática Remota (SPR) (en inglés *chats*), se han vuelto populares entre los usuarios del internet a partir de su lanzamiento, creando así día con día una mayor demanda y desarrollos de estos mundos. En estos ambientes en 3D los usuarios pueden escoger un personaje que los represente, denominado *avatar*, para navegar dentro del espacio que se ofrece y de esta manera mantener una conversación con otros usuarios y así establecer una comunicación más personalizada.

En el artículo *Exploring Virtual Worlds* Tom Furness [FURN91], explica que uno de los principales obstáculos a resolver en estos mundos es crear interfaces que utilicen habilidades humanas intuitivas, tales como la capacidad de crear gestos: *La meta principal es que las computadoras actúen como humanos en lugar del caso contrario*, y así de esta manera se puedan mejorar las relaciones humanas.

En la mayoría de los casos esta comunicación a través del teclado es con entradas en modo texto. Muchas interfaces permiten a los usuarios introducir lenguaje corporal y expresiones faciales al presionar alguna tecla o comandos especiales. Sin embargo, ésto es inconveniente debido a que los usuarios deben de teclear entradas extras para cambiar su expresión. En el mundo real la comunicación directa, *cara-a-cara*, entre usuarios no implica un razonamiento previo de emociones, sino la muestra de expresiones espontáneas y no verbales que se presentan en una conversación. Es un hecho que la comunicación no verbal es un aspecto muy importante del diálogo cara-a-cara.

Este tipo de comunicación otorga un modelo ideal para el diseño de una interface computadora-humano. Desde un punto de vista del modelado la cara se considera como un canal de comunicación que conjunta una serie de señales emocionales conversacionales y las codifica en despliegues faciales. Una de sus principales características es la multiplicidad de canales de comunicación.

Para llegar a simular un sistema como este es necesario estudiar como es que los humanos perciben la información, que tan sensitivo es hacia ella, y una vez que se realizó este proceso como es que la despliegan. La presencia de despliegues faciales en las interfaces humanas hacen la interacción hombre-máquina más eficiente mientras que diminuyen la carga de información necesaria. Es por eso que estas expresiones se consideran como señales de comunicación que mejoran la coordinación de las conversaciones.

Hasta el momento ninguno de estos ambientes gráficos ha incluido un avatar inteligente que pueda desplegar las diferentes emociones de un agente interno a estos mundos. El ejemplo más cercano es el denominado *Comic Chat*, en el cual hay que escoger la emoción que se desea por medio del ratón o mediante comandos especiales [DAME98].

El propósito del proyecto de interfaz para el sistema de pláticas remotas es desarrollar herramientas de reconocimiento de emociones a partir del procesamiento del lenguaje natural, de esta manera se tratarán de inferir las emociones que el usuario desea desplegar a partir del texto y mostrarlas en su representación. Esto evita el estar tecleando comandos innecesarios y aumenta el control del agente gráfico correspondiente. Como consecuencia el usuario sostiene una comunicación mas humana y natural.

6.2 Sistemas de Comunicación Remota

A finales de 1980 apareció uno de los primeros experimentos con ambientes gráficos de pláticas remotas en computadora, se trató de un sistema de conferencia y pláticas en línea llamado *WELL*. Muchas personas comenzaron a experimentar dentro de estos ambientes utilizando avatares como su representación. De esta manera nuevos amigos, matrimonios y muchas otras historias se pueden crear a través de este modo [DAME98].

Los mundos gráficos de pláticas remotas son los hijos de dos enormes tecnologías: la tecnología de comunicación basada en textos escritos y los juegos de computadora.

Estas fueron construidas alrededor de envio de mensajes textuales que son simples y fáciles de comprender. Como ejemplos de estos mundos se tienen *WELL*, sistemas que utilizan los llamados *MUDs* (Multi User Domains), *IRC* (Internet Relay Chat), y servicios en línea que proporcionan cuartos especiales para conferencias. Por otro lado experimentos tempranos que combinaron una interface gráfica con mensajes textuales comenzaron a mediados de los 80's con el sistema llamado *Habitat.* Los juegos de computadora fueron también una aportación importante debido a que se diseñaron para funcionar adecuadamente con un requerimiento mínimo de hardware, y así abrir camino para experimentar en los SPR. Con el lanzamiento de *Worlds Chat* en 1995 se vino una avalancha de tecnologías que promovieron la creación de estaciones en tercera dimensión llenas de efectos especiales y que tenían cientos de usuarios de internet disfrazados por sus propios avatares y manteniendo conversaciones de todo tipo en todo momento. Ejemplos de ambientes gráficos que merecen la pena de nombrar por ser bastante utilizados actualmente son *Worlds Chat Space Station, The Palace, Alpha World, Active Worlds, Worlds Awa, Virtual Places, Black Sun Passport, Comic Chat* y *Oz Virtual.* Toda persona puede tener acceso a estos mundos a través del internet y sumergirse durante horas de gran entretenimiento.

En la Figura 6.1 se muestra un ejemplo de los avatares que se obtienen en *The Palace* en el cual se puede platicar con otros usuarios y comunicarse por medio del texto escrito.

La manera en que estos avatares representan la entrada textual es a través de lenguaje natural demoninado no-verbal. Este lenguaje incluye el movimiento de labios, expresiones faciales, gestos y posicionamiento corporal. Sin embargo, hasta la fecha ningún sistema ha sido capaz de integrar la comunicación que se realiza con un sistema gráfico adecuado. La manera que *Comic Chat* realiza esto es por medio de un círculo de estados emocionales gráfico que permite cambiar la emoción al mover el ratón dentro del círculo como se puede observar en la Figura 6.2, cada vez que un usuario elige un caracter, este círculo de emociones le permite los cambios de ánimo según se quiera desarrollar la conversación. En este ejemplo se muestra a ANNA quien también se expresa a través de lenguaje corporal.

Como se ha puntualizado aún existen muchas características que añadir a estos mundos que las haga parecer una experiencia más real, como es el caso del despliegue de emociones. Existen una gran variedad de aplicaciones lucrativas que manejan este tipo de problemas. Por ejemplo la rama de la ingeniería dedicada al estudio de los Factores Humanos consiste en hacer que los usuarios se "sientan mejor"mientras que interactuan con interfaces de usuario por computadora. En

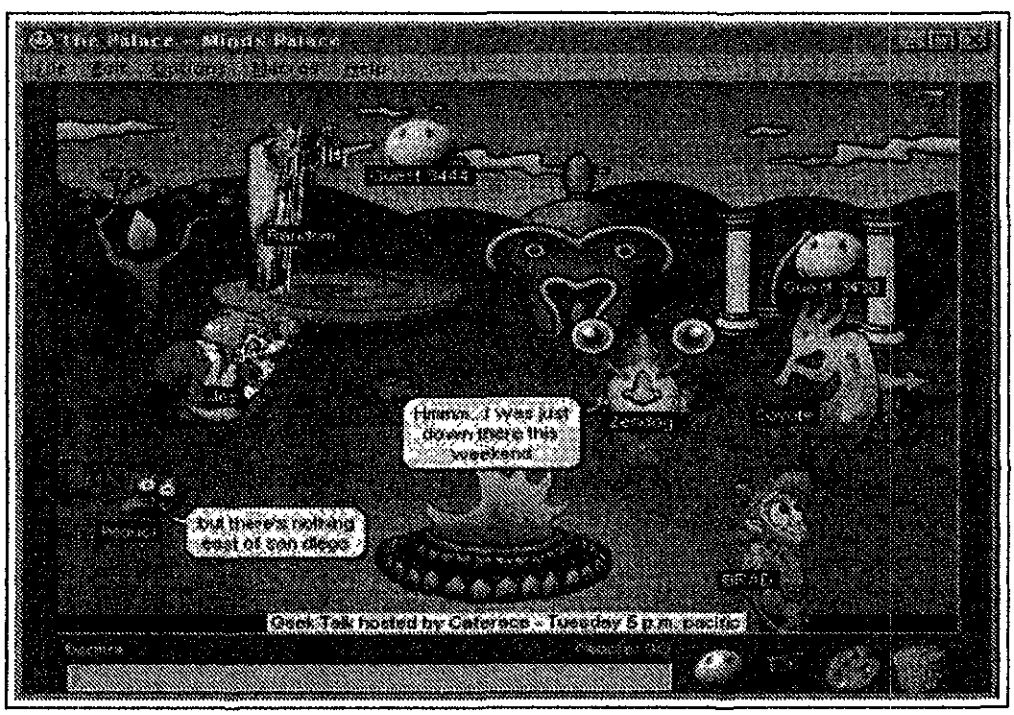

Figura 6.1: **Ejemplo de un Sistema de Pláticas Remotas en The Palace.**

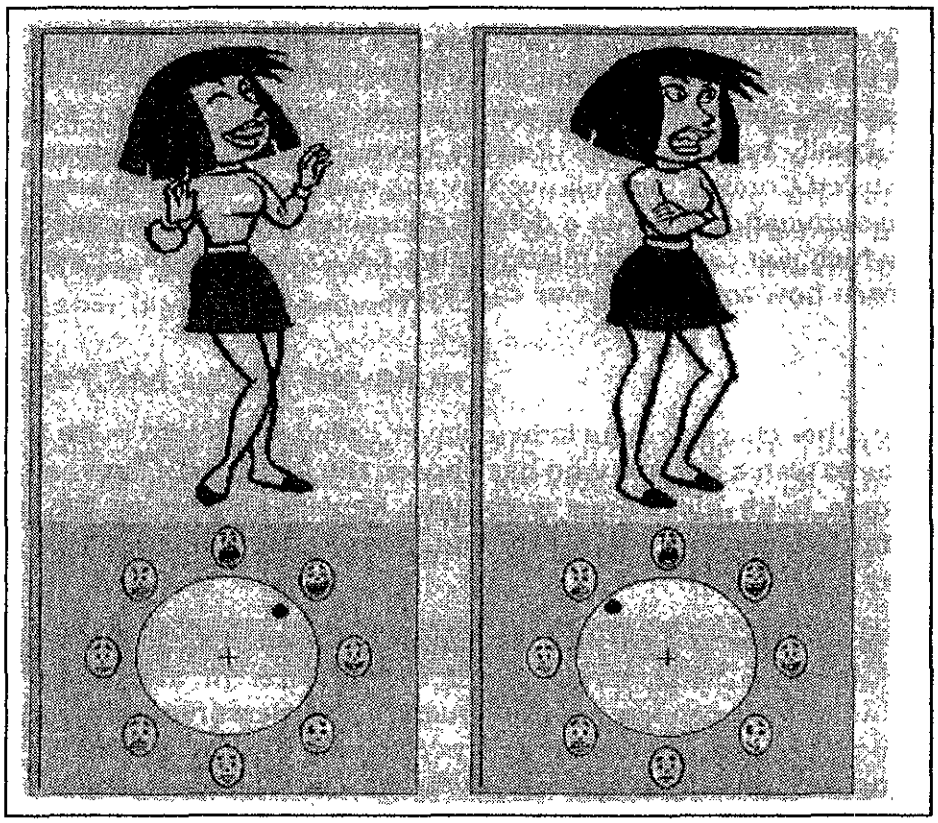

Figura 6.2: **Despliegues de emociones en ANNA, Comic Chat.**

este último campo se han reportado resultados de mayor productividad al tener interfaces hombre-máquina más amistosas, intuitivamente se puede concluir que un acercamiento emocional positivo facilita la interacción y aumenta la productividad. Es por eso que la investigación en el área de factores humanos ha incrementado y se desea poner mayor énfasis al trabajar con emociones humanas.

Para poder utilizar las emociones en sistemas gráficos de comunicación se requiere un sistema de clasificación de las mismas con mediciones apropiadas de intensidad. En la siguiente sección se analizarán con mayor detalle estos modelos emocionales y la importancia de su despliegue facial, así como una breve revisión de los intentos de clasificaciones emocionales que se han realizado.

Por otro lado es necesario obtener estas emociones a partir de un texto, lo cual implica añadir el uso de Interpretación del Lenguaje Natural (ILN). El diseño de un modelo emocional con ILN requiere de una descripción clara de las emociones fundamentales utilizadas.

En lo que resta de esta sección nos enfocaremos a describir los principios utilizados para crear la interfaz utilizada, la cual consiste de una animación facial por computadora y como se generan los despliegues emocionales.

6.3 Animación Facial por Computadora

Las interfaces de usuarios que fucionan como caracteres o agentes es un área emergente en el campo de la animación facial. Estos agentes o avatares deben poseer la habilidad de interactuar directamente con el usuario, con el fin de que estos agentes entiendan órdenes y hablen con el usuario en tiempo real y respondan con el mayor realismo que puedan. En este caso hablamos de interfaces sociales.

Uno de logros más dificiles es que la interface se comporte como humano en lugar de una simple computadora. Por lo cual se desea un humanoide con despliegue facial que realice este papel. Actualmente varios laboratorios académicos e industriales, como son el Medialab del MIT y la Universidad de California Santa Cruz trabajan con este tipo de prototipos.

Para desarrollar un modelo facial se deben determinar descripciones geométricas y capacidades de animación que representen las caras de interés. El rostro tiene una muy compleja y flexible superficie tridimensional, con variaciones de color y textura (como pueden ser las arrugas). Y de igual forma existen varias maneras de representar estas caras de manera que brinden animación efectiva. Una de las

formas más utilizadas en su creación es mediante superficies poligonales definidas por Frederic Parke [PARK96].

La mayor parte de los modelos faciales utilizan superficies poligonales debido a que la mayoría de las estaciones de trabajo tienen capacidad de desplegar superficies poligonales y pueden modificarse casi en tiempo real. Estas superficies se encuentran dentro de una red interconectada de polígonos como se puede observar en la figura 6.3 [PARK96] en la que se muestra el modelo canónico de T. Kurihara, el cual se constituye de alrededor de 3000 polígonos.

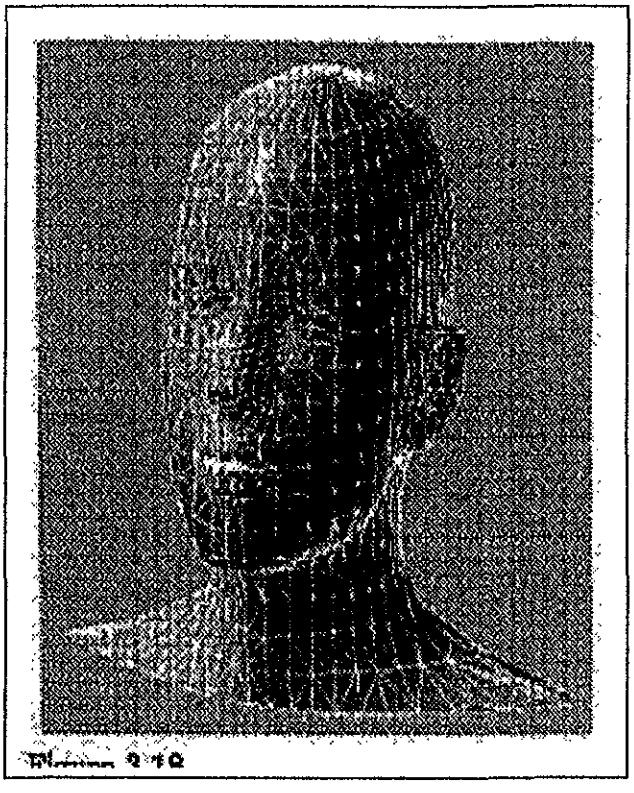

Figura 6.3: **Topología de polígonos de una cara [PARK96].**

Cuando se utilizan polígonos en una cara existen un cierto número de puntos a tomar en cuenta:

- Los polígonos se deben colocar de manera que permita a la cara flexionar y cambiar de forma naturalmente.

- Los polígonos deben aproximar la cara para cada expresión que se desea.
- La densidad de la información superficial se debe distribuir de manera que cree una curvatura natural en ella.
- Se debe utilizar el menor número de polígonos consistente con la obtención de resultados aceptables.
- Las aristas de los polígonos deben coincidir con ciertos bordes de la cara como son la orilla de los ojos, de los labios, la esquina de la boca. Así como también los bordes de los polígonos deben coincidir con las fronteras de la cara en las que se presenta un cambio de color, tales como son las cejas y los labios.
- Si las aristas son demasiado pronunciadas o visibles se aconseja suavizarlas para obtener una apreciación visual más natural.
- Como la cara es casi simétrica es suficiente con modelar un solo lado de ella, el otro lado se puede obtener haciendo un espejo o reflejando los polígonos en el plano de simetría.

La figura 6.4 tomada del Perceptual Science Laboratory de la Universidad de California Santa Cruz [PSLA99], muestra la topología poligonal sin cubrir de un despliegue facial, el cual al rellenar el área interna crea una imagen de la expresion facial en tercera dimensión.

6.3.1 Animación Facial

Existen al menos tres métodos utilizados para la animación facial: parametrización, interpolación y la basada en animación muscular. Su finalidad es manipular en el tiempo las superficies de la cara, de manera que tengan las expresiones deseadas en cada cuadro de la secuencia de animación. Este proceso modifica directamente los vértices de los polígonos o un punto de control de la superficie. A continuación se da una breve explicación de su funcionamiento.

Parametrización. Los esquemas de control de animación se pueden ver como un control de parámetros en el que se especifican y controlan los valores de éstos parámetros como función del tiempo. La animación facial se puede ver como dos actividades independientes: el desarrollo de parámetros de control e interfaces de usuario y el desarrollo de técnicas para crear la animación facial basada en esta parametrización. Hoy en día existen dos categorías principales de control de

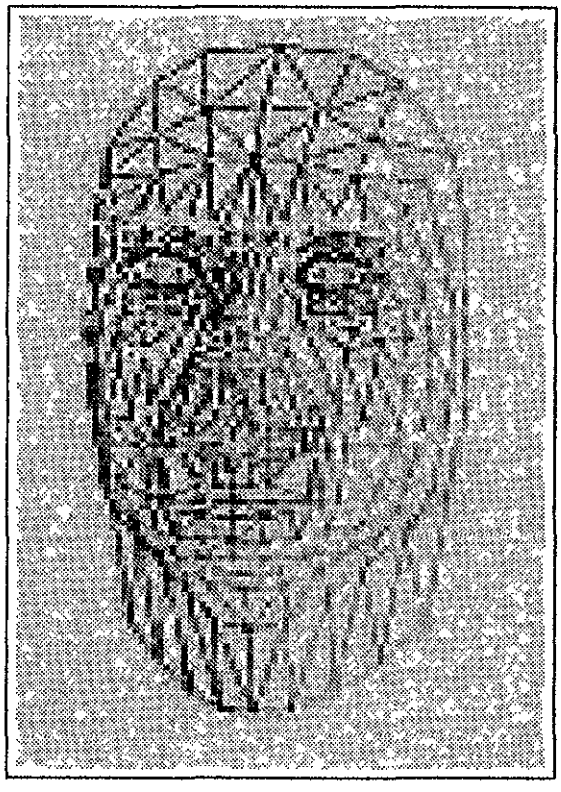

Figura 6.4: **Topología mostrando los polígonos constituyentes de la cara [PSLA99].**

parámetros. La más usada concierne el control de expresiones faciales y la segunda incluye el control de la forma facial o creación de una cara a partir de un universo de caras posibles.

Interpolación. Es muy importante para el caso en el que se crean caras nuevas o componentes de éstas a partir de otras previamente definidas. La noción de interpolación en el caso unidimensional se obtiene a partir de dos valores dados, de los cuales se determina un valor intermedio. El valor intermedio deseado se especifica por un coeficiente fraccional α:

$$x_{interp} = \alpha \times (x_1) + (1.0 - \alpha) \times (x_2) \qquad 0 < \alpha < 1.0 \tag{6.1}$$

donde:
α = coeficiente fraccional
x_{interp} = valor interpolado entre valor 1 y valor 2.
x_1= valor 1
x_2= valor 2

Este concepto básico puede ser fácilmente expandido a más de una dimensión aplicando el procedimiento en cada una de ellas. La idea se puede generalizar a superficies poligonales aplicando el esquema en cada vértice que define la superficie. Los vértices en cuestión tendrán dos posiciones tridimensionales asociadas, la inicial y la final. Y de esta manera formas intermedias se crean al interpolar cada vértice entre sus posiciones extremas. La interpolación no necesariamente debe ser lineal en el tiempo. El valor de α puede basarse en cosenos para dar este efecto no lineal [PARK96].

Con la interpolación se desea obtener de alguna manera datos geométricos que describan la cara al menos en dos diferentes poses de expresiones. Y después un parámetro de control (el coeficiente de interpolación), el cual se usa para cambiar la cara de una expresión a la otra en función del tiempo. Cuando se utiliza una topología de polígonos fija se manipula la posición de los vértices involucrados para así manipular la forma de la superficie facial. Para cambiar la expresión de una a otra en cuadros sucesivos, el punto de control de la superficie se modifica en una pequeña distancia. Y la posición de cada punto se determina al interpolar entre las posiciones extremas. La figura 6.5 muestra una transición creada entre dos expresiones. La imagen de en medio es la resultante de interpolar entre las dos laterales.

Figura 6.5: **Interpolación entre dos expresiones faciales**

Vectores Normales de Movimiento

Cuando se desea modificar más de un polígono a un mismo tiempo y cuyos vértices coinciden se pueden utilizar vectores normales de movimiento. Se calcula un vector normal a cada superficie poligonal y a partir de un promedio de éstos vectores vecinos, se puede obtener un vector normal correspondiente al vertice de unión de los mismos. Para encontrar la normal de un polígono, tome cualquiera de sus tres vertices $v1$,$v2$ y $v3$ del poligono. El producto cruzado:

$$[v1 - v2] \times [v2 - v3] \tag{6.2}$$

es perpendicular al polígono (este vector normalmente se normaliza). Después se promedian los normales de los polígonos adyacentes. Por ejemplo como se muestra en la figura 6.6 si n1, n2, n3 y n4 son los vectores normales a los polígonos y la normal del punto P se calcula promediando estos vectores y normalizándolos. El vector resultante se utiliza como el normal del punto P.

Una vez que se obtienen este vector es sencillo manipular un conjunto de polígonos interpolando valores que vayan de $-x$ a $+x$ donde x es cualquier valor tal que $x > 0$.

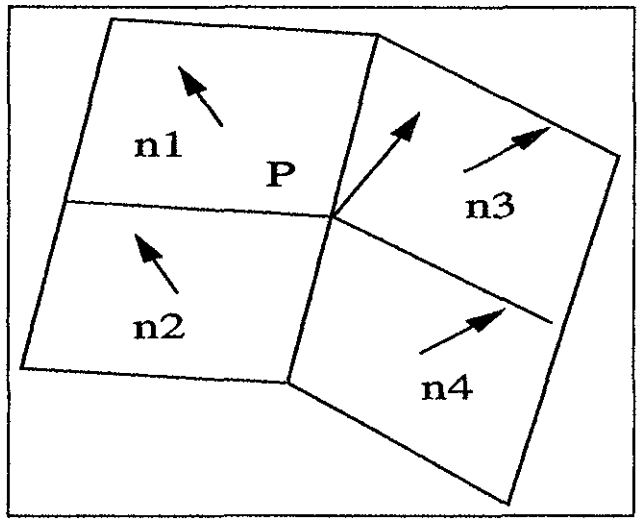

Figura 6.6: **Vectores normales a superficies poligonales**

Animación Muscular

La idea de este principio es el desarrollo de modelos con unos cuantos parámetros de control que emulen las acciones musculares básicas. Keith Waters desarrolló un modelo muscular que contrae y afloja los músculos [WATE93]. Sus músculos tienen propiedades vectoriales que los hacen independientes de la topología usada en la cara. Cada músculo tiene una zona de influencia. Los parámetros de control de este modelo se basan en el Sistema Codificado de Acciones Faciales (SCAF) en inglés *FACS*.

Hoy en día SCAF es uno de los sistemas más utilizados en el diseño y creación de despliegues faciales. Fue desarrollado por Paul Ekman y Wallace Friesen [EKMA78]. Este sistema describe las acciones básicas de los músculos faciales y su efecto en la expresión. Es bastante difundido porque divide toda la acción facial en unidades pequeñas llamadas unidades de acción (UA). Un ejemplo de una unidad es el músculo de la ceja interior que hace que las cejas se abran o cierren segun el caso. Cada unidad representa una acción muscular o una acción de varios músculos pequeños. En total existen 66 unidades clasificadas cuya combinación genera diversas expresiones.

SCAF sólo se ocupa de describir movimientos faciales. Es ampliamente utilizado en animaciones por computadora pues especifica una forma de controlar el movimiento facial a través de acciones musculares, necesarias para generar los cambios en la expresión. SCAF se limita a aquellos músculos cuyo movimiento es voluntario, por

lo que no incluye todos los músculos de la cara aunque si los necesarios para dar el efecto de realismo.

6.3.2 Tratamiento de la cara como una imagen

Las imágenes creadas en computadora se pueden manejar como dos arreglos rectangulares cuya unidad es el pixel. Este tiene un valor asociado de color que se despliega en el monitor. Los rangos de valor de estos colores pueden ir desde unos pocos bits hasta 36 bits. Y por lo general se representan como tripletas de 8 bits cada uno, 8 bits para cada uno de los tres componentes primarios: rojo, azul y verde [GONZ90].

Dentro de la síntesis de imágenes se pueden realizar 3 tareas:

- transformar el modelo geométrico y sus componentes en un sistema de coordenadas observable.
- determinar que superficies son visibles y
- procesar el valor de los colores para cada pixel de la imagen basado en las condiciones de luz.

Para obtener movimientos completos de la cara, como rotación y traslación de coordenadas, los componentes de la imagen se definen en un sistema de coordenadas a las que se aplican las transformaciones deseadas. Estas transformaciones se pueden definir como un conjunto de ecuaciones aplicables al sistema de coordenadas. Por ejemplo un sistema tridimensional con sus puntos definidos en base a las coordenadas (X,Y,Z), se desea desplazar (Xo,Yo,Zo). Esto se realiza empleando las ecuaciones que realizen la operación deseada:

$$X' = X + Xo \tag{6.3}$$
$$Y' = Y + Yo \tag{6.4}$$
$$Z' = Z + Zo \tag{6.5}$$

Sin embargo es más conveniente expresar estas operaciones con notación matricial que facilite el procesamiento, por ejemplo:

$$\begin{pmatrix} X^* \\ Y^* \\ Z^* \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & Xo \\ 0 & 1 & 0 & Yo \\ 0 & 0 & 1 & Zo \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Donde X^*, Y^*, Z^* son las coordenadas del punto trasladado. Típicamente, en lugar de coordenadas tridimensionales se utilizan coordenadas de 4 dimensiones. Entonces el punto (X,Y,Z) se transforma en (X,Y,Z,1) y a estas se les conoce como coordenadas homogéneas. Las coordenadas homogéneas dan una serie de ventajas como son el que todas las operaciones puedan expresarse en matrices de 4X4 lo que facilita su procesamiento además se pueden realizar operaciones múltiples conjuntamente. Tal es el caso de las operaciones de rotación, traslación y escalamiento. A continuación se especifican las matrices necesarias para la operación de escalamiento:

$$\begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

En donde S es el factor de escalamiento. Y rotación:

$$\begin{pmatrix} \cos\theta & \theta & 0 & 0 \\ -\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

En este último caso la rotación de un ángulo θ afecta únicamente a los valores de las coordenadas X y Y.

Por lo tanto, las partes de un modelo geométrico se definen en sistemas coordenados, y a éstos se les aplican transformaciones que generan el modelo completo, con cambios de tipo: rotación de la cara, traslación de los ojos o de la cara completa así como escalamiento.

Para el caso de la aplicación de color y textura se utiliza la técnica de Gouraud. En esta técnica se otorga un color a cada vértice del polígono, y los valores intermedios de color de un vértice a otro se obtienen usando interpolación lineal. El valor de color se basa en un vector de superficie para el vértice correspondiente. Y por lo general el vector de color se crea sumando los vectores normales para todos los polígonos que comparten el vértice. Esta técnica elimina discontinuidades de sombreado entre polígonos adyacentes y la superficie aparece suavizada.

6.3.3 Iluminación y color

En el modelo de luz y sombras de **OpenGL** (lenguaje de programacion utilizado para este trabajo) una superficie de color se determina para cada vértice del polígono. Los colores entre los vértices son determinados usando el método de Gouraud [DAVI97]. Y la superficie del color de cada vértice es la suma de 3 componentes: un componente del color establecido para la superficie del polígono llamado término de emisión, el componente de la luz del ambiente global y la suma de contribuciones de todas las otras fuentes de luz especificadas. La suma se especifica como:

$$color_{vert} = termino_{em} + ambiente_g + \sum luz_{otras} \tag{6.6}$$

donde:
$color_{vert}$ = Color del vértice
$termino_{em}$ = Término de emisión de la superficie del polígono
$ambiente_g$ = Componente de luz en el ambiente global
luz_{otras} = Contribuciones de otras fuentes de luz especificadas

Cada uno de estos componentes tiene contribuyentes de los colores primarios: rojo, azul y verde. Y el resultado final se limita al rango de [0,1]. Este modelo no incluye reflexiones de un objeto a otro.

Utilizando estas técnicas se constituyó el módulo de animación del sistema de expresiones faciales que más adelante se describe en el capítulo 8. Este modelo se tomó del desarrollo creado para el DECface de una cabeza parlante [WATE93], y constituye el módulo de visualización y animación en la presente tesis.

Parte II

Agentes Inteligentes

Capítulo 7

Agente con Reconocimiento de Guiones

7.1 Objetivo del Sistema

Se propuso crear un desarrollo de reconocimiento de guiónes. Estos guiónes deben otorgar a un agente de software características que permitan la simulación de agentes inteligentes que funcionen continuamente de una manera autónoma con el fin de aprender de su experiencia propia.

7.2 Modelo del Sistema de Desarrollo

Como se explicó en el capítulo 3 existen ciertas características que se deben tomar en cuenta para la creación y desarrollo del comportamiento de un agente inteligente. Estas características son: reactividad, autonomía, comportamiento colaborativo así como capacidad de inferencia, movilidad y adaptación (refiérase al capítulo 3 de Agentes). Para poder otorgar todas estas cualidades se necesitó de un sistema experto que conjuntara el conocimiento necesario para reconocer situaciones diarias y que contará con cierta capacidad de desición.

En este proyecto se desarrollaron programas de software que utilizan guiones para el reconocimiento de situaciones cotidianas. Para probar su funcionamiento correcto se utilizó el sistema de la casa asistida por un robot utilizado por Jesús Savage [SAVA95]. En este caso nuestro agente de software es el robot. Sin embargo, estas técnicas pueden utilizarse en cualquier otro desarrollo que incluya agentes y

que posean las características definidas anteriormente.

El agente de software representado por un robot ayuda en las tareas dentro de la casa. Este agente recibe órdenes y las ejecuta. Existe un número de objetos en la casa que el robot puede utilizar. Los comandos son introducidos a través de texto y ejemplos de estos son: "Robot dale los zapatos al padre", "Robot trae al perro". Y una vez que el robot comienza a ejecutar una acción no permite que ninguna otra interfiera hasta que se termina.

La Figura 7.1 muestra el diagrama a bloques del sistema utilizado.

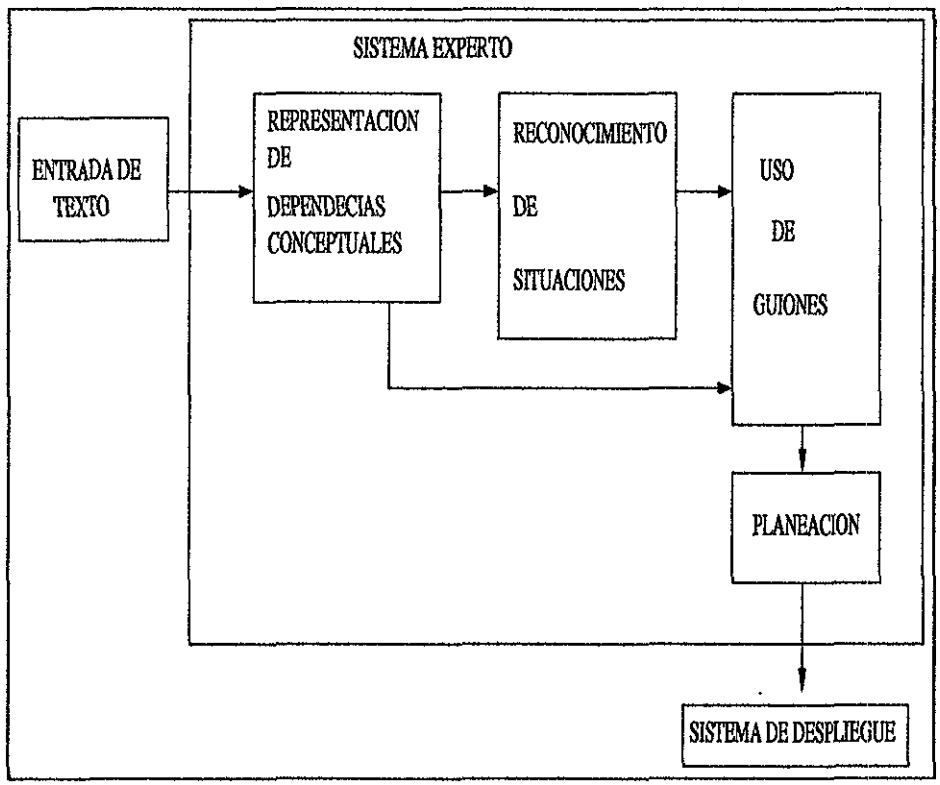

Figura 7.1: **Diagrama a Bloques del Sistema de Guiones.**

Las órdenes se toman a partir de una entrada de texto, las cuales para su reconocimiento se transforman en Dependencias Conceptuales que en realidad son primitivas operacionales (refiérase al capítulo 4 de procesamiento del lenguaje natural). Una vez que se crean estas Dependencias Conceptuales se pasa a una etapa de reconocimiento de situaciones definidas que utiliza Guiónes. Y por ultimo la

ejecución de estas órdenes conlleva una etapa de planeación que se encargue de administrar las tareas y recursos que lleven el agente a su objetivo final. Todo esto se visualiza en un módulo de despliegue. Todas las etapas anteriores al despliegue se implementaron por medio de un sistema experto.

Este sistema experto se desarrolló principalmente en CLIPS que se explica en la siguiente sección. También se utilizaron rutinas de comunicación en C a través de sockets de la red, que permiten la comunicación a traves del Ethernet.

7.2.1 CLIPS

El sistema experto que se utilizó para el desarrollo de las aplicaciones mostradas en este y el siguiente capítulo fue principalmente CLIPS, Lenguaje C Integrado a Sistemas de Producción, el cual es un ambiente completo para desarrollar sistemas expertos. CLIPS fue diseñado por la NASA con el propósito de otorgar alta portabilidad, bajo costo y una fácil integración con sistemas externos. Fue escrito utilizando lenguaje de programación en C para facilitar estos objetivos [CLIP93].

Originalmente la primera metodología utilizada en CLIPS fue a través de reglas con encadenamiento hacia adelante basadas en el algoritmo Rete, estos conceptos se explican en el capítulo 2. CLIPS posee tres elementos básicos: una lista de hechos que contienen datos representando el estado actual del "mundo"; una base de conocimientos basada en reglas *IF-THEN*; y una máquina de inferencias. La porción *IF* corresponde a una serie de condiciones preestablecidas que especifican los hechos (datos) por los que la regla se aplicó, y la porción *THEN* de la regla es el conjunto de acciones a ser ejecutada cuando se dispare la regla. Los hechos y las reglas son llamadas productos y la colección de condiciones y acciones a ser tomadas son construidas en una red de reglas llamada sistema de producción. Usando el algoritmo Rete la máquina de infrencias encuentra la correspondencia entre los hechos que determinan que reglas deben ejecutarse y cuando.

Cada regla en CLIPS se conforma de 3 partes: la primera es el nombre de la regla, la segunda representa los hechos que la activen y la tercera representa las acciones a ser ejecutadas. Por ejemplo tenemos el siguiente pseudocódigo.

IF está lloviendo
THEN use un paraguas

se representa en CLIPS como:

```
(defrule lluvia-paraguas
   (clima lloviendo)
   =>
   (assert (use paraguas))
)
```

En este ejemplo se puede observar que el nombre de la regla es lluvia-paraguas, el hecho que la dispara es (clima-lloviendo) y el commando assert es el que crea el nuevo hecho (use paraguas). Esta representación de reglas es una manera sencilla de codificar el conocimiento eficientemente. Y como se dijo antes, cada regla representa una porción del conocimiento que a su vez promueve que las reglas interactúen una con otra a través de los hechos. Para ampliar más sobre el tema de CLIPS refiérase al Apéndice A de esta tesis.

7.3 Descripción del Sistema

Como se dijo la representación de las órdenes o acciones se representó utilizando Dependencias Conceptuales definidas en el capítulo 3. Sin embargo aún falta definir como se representarían los objetos y otros agentes que se encuentren en el mismo entorno que nuestro agente de software el robot.

7.3.1 Representación utilizando Objetos

Anteriormente se dijo que CLIPS es un lenguaje orientado a objetos, y en la simulación utilizada todos los entes individuales y herramientas se representaron como objetos. Cada ente pertenece a una clase de objetos y cada clase posee distintos atributos que son heredables. Un ejemplo de la representación general del objeto ACTOR posee atributos o campos de información del mismo objeto, esto se muestra en la tabla 7.1:

Campos del objeto ACTOR	función del atributo
ID	identificador
LOCATION (*localización*)	se refiere al lugar dentro de la casa
ACTION (*acción*)	puede ser que este trabajando o descansando
CARRYING(*cargando*	que objetos ha recogido de su ambiente
X	localización de la coordenada x dentro de la casa
Y	localización de la coordenada y dentro de la casa
Z	localización de la coordenada z dentro de la casa

Tabla 7.1. Campos de información del objeto ACTOR.

Como se observa en la estructura ACTOR, se tienen diversos campos para representar sus características. Cada uno posee un identificador; un lugar de localización que especifique en que cuarto se encuentran localizados los actores; *cargando* contiene las herramientas que el actor se encuentra cargando; *usando* es lo que el actor esta usando en su persona como un sueter, pantalónes; *y x, y, z* dan las coordenadas del actor dentro de la casa. Ciertos campos como *cargando* puede contener múltiples valores, por ejemplo un mismo actor puede cargar un jabón y una toalla a un mismo tiempo. Esta clase de objetos puede crear varias instancias directamente así como activar una regla en el lado derecho. Un ejemplo de la clase ACTOR es:

*(***ACTOR** *Robot (ID Robot)(LOCATION sala-comedor)(X 280)(Y 330)(Z 0))*

Cuando alguno del los compartimentos no tiene asignado un valor toma el que se especifica en el campo por default o simplemente no posee ningún valor.

Aquellas entidades o instrumentos sobre las cuales los Actores pueden ejecutar alguna acción son designadas como objetos inanimados ó INANIMATE. Un ejemplo de su representación es:

*(***INANIMATE** *libro (ID libro)(LOCATION recamara-padres)(X 280)(Y 32))*

Como se observa sus compartimentos son similares a los de ACTOR solo que también poseen campos extras denominados: *dueño* el cual especifica a quien pertenece y *atributo* especifica características del objeto como son color, forma etc.

Para cada objeto que forma parte del sistema se creo una representación con atributos únicos y sobre éstos objetos se ejecutan acciones definidas por las Dependencias Conceptuales (DC). Por ejemplo si se quiere que el robot que vaya a la

cocina se ordena "Robot ve a la cocina"el cual genera la siguiente DC:

*(***PTRANS***(ACTOR Robot)(OBJECT Robot)(FROM lugar_robot)*
(TO cocina))

Esta dependencia a su vez, busca y activa una regla que encuentre el lugar en que se localiza al robot. Ya que se desencadenó la orden entonces se realiza el movimiento físicamente.

También se crearon rutinas de graficación escritas en C para observar los cambios o movimientos físico del robot. Estas rutinas se denominan CONDOR [SAVA95].

Todas las acciones que desencadenan las dependencias conceptuales son analizadas y ejecutadas através del modulo de planeación. Este módulo genera otras dependencias conceptuales a partir de la primera para completar la información faltante. Y estas nuevas dependencias disparan nuevas reglas las cuales crean nuevos hechos. En el caso del ejemplo anterior, el módulo de planeación se encarga de encontrar el lugar en que se encuentra el robot disparando nuevas reglas para esto. Y como resultado se obtiene una nueva dependencia conceptual con los datos faltantes:

*(***PTRANS***(ACTOR Robot)(OBJECT Robot)(FROM sala)(TO cocina))*

7.3.2 Uso de STRIPS

Como se mencionó en el capítulo 4 los guiones denominados STRIPS se constituyen de: todos los hechos en el modelo original, menos todos los hechos en la lista de elementos a borrar, más todos los hechos de la lista de adición. Para mostrar la manera en que se implementó esto en el sistema se explica a continuación un ejemplo.

Suponga que el robot se encuentra en la cocina y la pelota en la sala. Codificando en DC's los pasos a seguir del ejemplo se tienen las siguientes DC's generadas con sus operaciones necesarias para esta orden:

*(***PTRANS***(ACTOR robot) (OBJECT robot) (FROM cocina) (TO sala))*
Para ejecutar este PTRANS se generan primitivas más sencillas a través de una planeación para ir de un cuarto a otro a través de la ruta más corta. Estas primitivas tienen la forma de:

(Encuentra-ruta-corta cocina sala)
Una vez que se ejecutó este proceso se procede a realizar las demás dependencias

conceptuales.

*(***GRASP***(ACTOR robot) (OBJECT pelota) (FROM suelo) (TO robot))*
*(***PTRANS***(ACTOR robot) (OBJECT robot) (FROM sala)*
(TO cuarto del niño))

Aquí de nuevo se generará otra primitiva para encontrar la ruta más corta:
(Encuentra-ruta-corta cocina sala)

y finalment se le dará la pelota a un actor, en este caso la madre.

*(***ATRANS***(ACTOR robot) (OBJECT pelota) (FROM robot) (TO madre))*

STRIPS se ayuda de diagramas para mostrar el orden de ejecución y generación de nuevas instrucciones. Un ejemplo utilizando el diagrama para esta orden se muestra en la figura 7.2, donde cada renglón contiene la aplicación de una DC u operación. Las casillas guardan las salidas y hechos generados como son: la situación actual del robot, situación actual de la pelota (también puede checar si es que la tiene en sus manos o no).

La primera columna consiste de todos los hechos en el estado de inicio que son necesarios para que se suscite o se les aplicará la operación. Las segunda columna contiene los hechos que no se borraron al aplicar la operación así como los nuevos hechos añadidos debido a la operación. Enmedio de las dos se muestra la operación ejecutada sobre estas condiciones presentes. Por lo que cada renglón representa el estado en un cierto tiempo t y el renglón siguiente se refiere al estado en el tiempo $t+1$ después de haber aplicado el operador.

Se puede observar de la figura 7.2 que en el nivel derecho de la tabla existen una serie de hechos a los cuales se les aplican un conjunto de operaciones que cambian el estado actual del mundo. Al aplicar una operación de **PTRANS** en el robot se generarán una serie de nuevos hechos que quedan en la casilla localizada inmediatamente a la derecha de la instrucción. En este caso los nuevos hechos generados fueron:

lugar robot = sala
lugar pelota = sala

Algunos de los hechos anteriores en el renglón 1, se borran debido al cambio de estado generado por el **PTRANS**. Y únicamente los hechos que no se borraron continúan en la segunda columna, en este caso se trata de las instancias:

pelota
robot
lugar pelota= sala

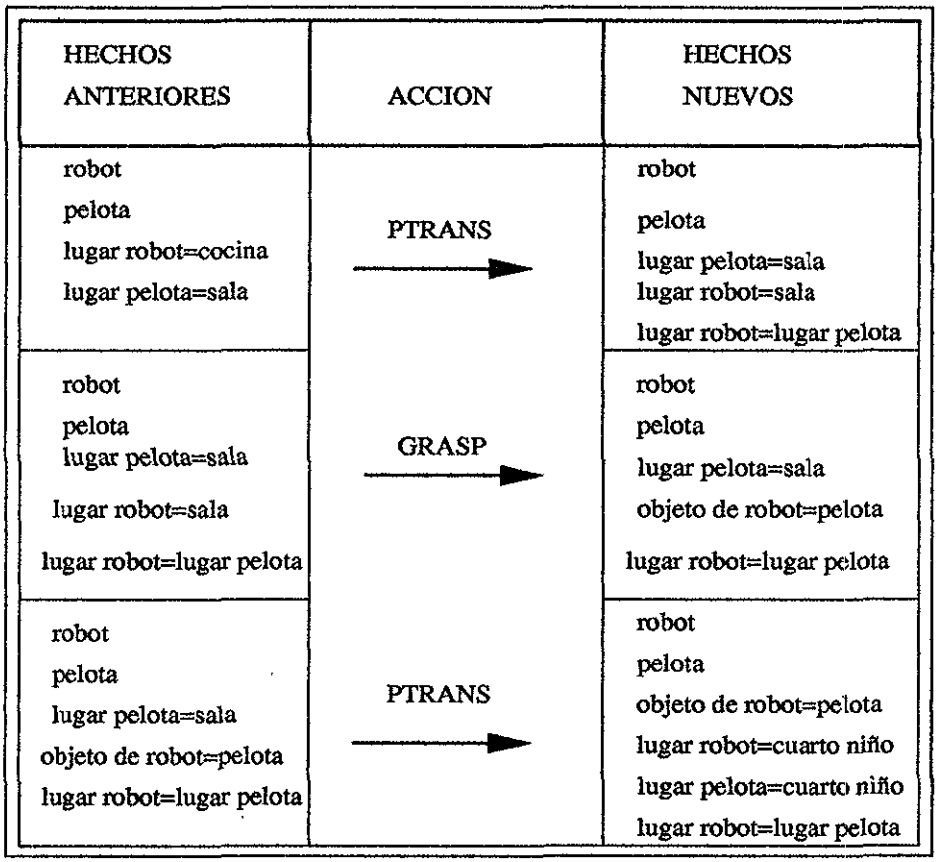

Figura 7.2: **Diagrama generado por la orden: robot lleva la pelota al cuarto del niño.**

Este proceso se realizó sucesivamente al aplicar una nueva instrucción, en este caso la siguiente fue **GRASP**. Por lo tanto los nuevos hechos se agregan a la derecha de la instrucción recien ejecutada. Y permanecen únicamente los que no se modifican con la nueva instrucción.

Con STRIPS se trató de generalizar al máximo todos los hechos generados, así como la operación que se ejecutó sobre estos. Una generalización sería el hecho de poder aplicar estas operaciones a cualquier objeto que se traslade de un lugar a otro, y el resultado final es el mostrado en la figura 7.3.

Una vez establecida la generalización, estas operaciones se guardaron en la instancia del STRIP (lo cual se explica en la siguiente sección), cargándola a la base

HECHOS ANTERIORES	ACCION	HECHOS NUEVOS
actor1 objeto1 lugar actor1=lugar1 lugarobjeto1=lugar2 lugar1 distinto de lugar2	PTRANS →	actor1 objeto1 lugar objeto1= lugar2 lugar actor1 = lugar objeto1
actor1 objeto1 lugar objeto1= lugar2 lugar actor1 = lugar objeto1	GRASP →	actor1 objeto1 lugar objeto1= lugar2 lugar actor1 = lugar objeto1 objetode actor1=objeto1
actor1 objeto1 lugar actor1 = lugar objeto1 objetode actor1=objeto1	PTRANS →	actor1 objeto1 lugar actor1 = lugar objeto1 lugar objeto1= lugar3 lugar actor1 = lugar3 lugar3 diferente de= lugar1 y lugar2

Figura 7.3: **Generalización del STRIP anterior.**

de datos del robot y conformando un guión. Este guión contiene todas aquellas acciones próximas a ejecutarse menos las que ya se ejecutaron y borraron.

Observe como al generalizar el robot pasó a ser un actor que puede tomar otra identidad diferente. Pelota paso a ser un objeto cualquiera que se quiere trasladar de un cuarto a otro. Y la sala, cocina y cuarto del niño pasó a ser cuarto1, cuarto2, y cuarto3, obteniendo un guión generalizado que mueva objetos de un lugar a otro. Un ejemplo final del guión general codificado es:

Guión mover-objetos {

actor =*actor1*
objetos =*objeto1*
acciones (o dependencias conceptuales que actuan)=
*(***PTRANS***(ACTOR actor1) (OBJECT actor1)*
(FROM cuarto1)(TO cuarto2))
*(***GRASP***(ACTOR actor1) (OBJECT objeto1) (FROM cuarto2)*
(TO actor1))
*(***PTRANS***(ACTOR actor1) (OBJECT actor1)*
(FROM cuarto2) (TO cuarto3))
}

Los nombres en itálicas de *actor1, cuarto1, cuarto2 y cuarto3* son variables que toman valores específicos al ejecutar el guión. Este guión se guarda en memoria y en el momento en que se genere una instrucción como: "Juan, lleva el correo al cuarto de tu padre", se inicia el guión de llevar objetos porque se infirió que se está moviendo un objeto de un lugar a otro. Y el guión toma los valores adecuados sustituyendo actor1 por Juan que es quien ejecutará la acción, objeto1 por correo que es el objeto sobre el que se ejecuta la acción y también se obtienen los valores a donde se tiene que mover. El guión generado ahora queda como:

Guion mover-objetos {

actor = Juan
objetos = correo
acciones =
*(***PTRANS***(ACTOR Juan) (OBJECT Juan) (FROM sala)*
(TO lugar-correo))
}

Recuerde que también se localiza el lugar del correo. Supongamos que el resultado es que el correo se encuentra en la entrada de la casa. Nuevamente se genera una primitiva que muestre la ruta más corta para ir de la sala, lugar donde está Juan, a la entrada de la casa.

*(**GRASP**(ACTOR Juan) (OBJECT correo) (FROM lugar-correo) (TO Juan))*
*(**PTRANS**(ACTOR Juan) (OBJECT Juan)*
(FROM entrada-casa) (TO cuarto-padre)))}

Los guiones (STRIPS) utilizados para este sistema se crearon usando representación con Dependencias Conceptuales (DC). Es decir, las acciones se representarón utilizando y generando nuevas DC's, que en su globalidad forman los guiones [SCHA81].

7.3.3 Reconocimiento de situaciones

El contexto juega un papel importante para reconocer la informacion faltante. El sistema experto posee una base de datos que maneja el estado actual de todo objeto presente en el mundo gráfico. Y cada vez que se pide una nueva orden se checa esta base de datos y se procede a ejecutar las nuevas órdenes. Si la orden pertenece a una situación conocida anteriormente entonces el robot realiza una secuencia de acciones (que conforman una tarea) en forma automática.

Al hacer los módulos de generación de situaciones se utilizó una etapa de entrenamiento con la cual se va enseñando al robot cada una de las fases a desarrollar. En ésta etapa se crea un objeto que corresponde a una clase general utilizada para todos los guiones, la cual se va llenando con las acciones que se ordenan. En esta etapa de aprendizaje el sistema inhibe toda otra acción ajena al aprendizaje.

Ahora, el problema que aparece es como reconocer la situación adecuada dadas las condiciones presentes. Normalmente comienza cuando el usuario genera una sentencia relacionada o el robot ejecuta ciertas acciones que pertenecen al patrón de esa situación. En ambos casos se crean dependencias conceptuales que coinciden con aquellas pertenecientes a una situación dada. Por ejemplo si se ordena "Robot let's give a shower to the dog". El robot, dentro del ambiente gráfico, debe moverse del lugar donde se encuentre a donde se supone se encuentra el perro (que implica una búsqueda), llevarlo al baño y bañarlo. Si las instrucciones de esta acción corresponden en más de un 50%, en este momento la situación correspondiente como

puede ser "busca_objeto" se dispara y ejecuta, creando nuevas instrucciones primitivas como buscar la ruta más óptima para encontrar al perro. De igual manera durante el proceso se borran hechos como lo especifica Fikes y Hart en su teoria de STRIPS [HUTC95].

Como se dijo las operaciones correspondientes a cada situación se almacenan con un formato similar al de las DCs. En el reconocimiento se decodifican estas operaciones nuevamente pero ahora utilizando el contexto adecuado. Es decir, pueden cambiar la localización de los actores, los objetos y los mismos actores. A continuación se muestran ejemplos de una situación codificada de las dependencias conceptuales.

(**PTRANS***(ACTOR actor) (TO lugar-del-objeto))* (buscar un objeto)
(**GRASP***(OBJECT objeto))* (tomar un objeto)
(**PTRANS***(ACTOR actor)(TO bathroom))* (trasladar el objeto al baño)

Esta acción no solo implica el llevar el perro al lugar adecuado sino también los utensilios necesarios para completarla como son el jabón y la toalla, lo que generaría las siguientes instrucciones.

(**PTRANS***(ACTOR actor)(TO lugar-del-objeto))* (buscar el jabón)
(**GRASP***(OBJECT objeto))* (tomar el jabón)
(**PTRANS***(ACTOR actor)(TO lugar-del-objeto))* (buscar la toalla)
(**GRASP***(OBJECT objeto))* (tomar la toalla)

Estas operaciones se almacenaron en una estructura objeto del guión creado. La etapa de reconocimiento se maneja en todo momento que no se encuentre activa la etapa de entrenamiento. Esto se logra monitoreando constantemente cada DC que se crea.

Según se iban ordenando tareas y generando dependencias conceptuales, éstas se codificaron de la manera mencionada anteriormente, para formar parte de un nuevo guión de reconocimiento. Este guión de reconocimiento se comparó contra los demás ya existentes. En el momento en que se tuvo un 50% de similitud en las etapas se disparó una bandera con el guión identificado. No sólo se compararon las dependencias conceptuales codificadas, sino también deben coincidir, al menos en el porcentaje indicado, los objetos de las mismas así como la localización de donde se generó la actividad. Al activarse la bandera, si aún existen etapas que no se han realizado, éstas se deberán completar si es posible de manera que se cumpla al 100%. Es decir, si aún quedan algunas acciones por cumplir se generan automáticamente.

Un ejemplo de un guión general junto con sus campos se muestra en la siguiente tabla 7.2:

Guión de bañar el perro	**Posible situación**
identificador	buscar un objeto
actor	robot
objeto	perro
objetos adicionales	objeto2=jabón
	objeto3=toalla
receptor	ninguno
posición inicial	sala
fases	(PTRANS (ACTOR actor)(TO lugar_del_objeto))
	(GRASP (OBJECT objeto))
	(PTRANS (OBJECT objeto)(TO baño))
	(PTRANS (ACTOR actor)(TO lugar-del-objeto2))
	(GRASP (OBJECT objeto2))
	(PTRANS (ACTOR actor)(TO lugar-del-objeto3))
	(GRASP (OBJECT objeto3))
operando	verdadero

Tabla 7.2. Ejemplo de un objeto para un situación.

La representación en CLIPS de este objeto se encuentra en el Apéndice A.3.

Se consideró que un guión posee 8 etapas como máximo que se representan con Dependencias Conceptuales. Por lo tanto mientras que no se reconozca ninguna situación y se llene el guión de reconocimiento con 8 etapas, el mismo se irá vaciando en un formato del primero que entra es el primero que sale (*FIFO*).

Una vez que se reconoció la situación, el sistema comienza a decodificar las fases del guión identificado. Y se realiza en inversa el proceso de decodificación de las DCs, generando las dependencias conceptuales faltantes a ejecutará del sistema experto.

También existe la ejecución del guión completo. Esto sucede al momento de mencionar las palabras de identificación del mismo. Este se ejecuta completamente hasta finalizar la última acción.

Capítulo 8

Reconocimiento de Emociones

8.1 El Objetivo del Sistema

Se propuso el crear un sistema inteligente que infiera las diferentes emociones rastreándolas de una entrada tipo texto, esto con el propósito de aplicarlas posteriormente en los SPR de manera que los avatares muestren diferentes estados de ánimo.

Un procesamiento correcto es dificil de realizar para entender el lenguaje natural. Sin embargo en este caso no se esta analizando el texto para entender lo que los usuarios estan escribiendo, sino para inferir las emociones que desean mostrar.

8.2 Modelo Emocional

Debido a que la finalidad del sistema es el de utilizarlo en conversaciones sostenidas a través de la red es necesario que se posean las dos siguientes entidades:

- Agente, entidad encargada del despliegue emocional de la persona que se encuentra tecleando su mensaje.
- Receptor, es el agente al cual se le comunica el despliegue emocional.

Para cumplir con el objetivo del proyecto fue necesario tomar ciertos requerimientos en consideración. Se necesita que el sistema posea las siguientes características:

1. Un conjunto inambiguo y conciso de categorías emocionales que especifiquen un valor de intensidad numérica. Definidas anteriormente por los conjuntos difusos.

2. Una función de transición para modelar las perturbaciones del estado emocional del sistema, la cual debe de funcionar de manera independiente para cada categoría de emociones pues no siempre todas las emociones van a sufrir cambio, bajo un mismo estímulo.

3. Una forma de caracterización para el agente transmisor y el receptor, y el modelo emocional de ambos entes incluídos dentro de la comunicación.

4. Una manera de establecer la información acerca de los efectos emocionales que deberá de ser especificada en forma de estructuras de datos simples o enunciados claros de lenguaje natural. Por ejemplo considere el enunciado: "Si un usuario esta irritado, en la próxima perturbación molesta, cambiará la emoción de un estado irritado a enojado".

Además de intentar cubrir todas estas características, es obvio que se espera que el sistema sea sencillo de utilizar por el usuario y con la posibilidad de modificaciones futuras.

8.3 Descripción del Sistema

La figura 8.1 muestra los componentes del sistema conversacional como entidades individuales. El primer paso es cuando existe una entrada de texto y se aplica un procesamiento del lenguaje. En este módulo se realiza el reconocimiento de palabras claves y emoticons de la manera descrita en la siguiente sección. Después de esto el sistema experto formula una hipótesis de emoción, la cual incluye la función de transición exponencial y el proceso de lógica difusa. La salida de este módulo genera una cuantificación de la emoción dentro de los conjuntos que se definieron, para que estos valores entren directamente al sistema de animación a cargo del despliegue facial y control del agente.

Se utilizó una clasificación (definida en el capítulo 5) basada en el modelo propuesto por Izard [IZAR77] el cual define 7 estados fundamentales: interés, felicidad, sorpresa, tristeza, enojo, miedo y disgusto.

Como se mostrará en la siguiente sección a existen señales explícitas para reconocer estos estado emocional a partir de una entrada textual. Al momento de mencionar las palabras de identificación de cierta emoción se ejecuta el cambio sobre el avatar (en caso necesario) hasta finalizar y llegar al despliegue deseado.

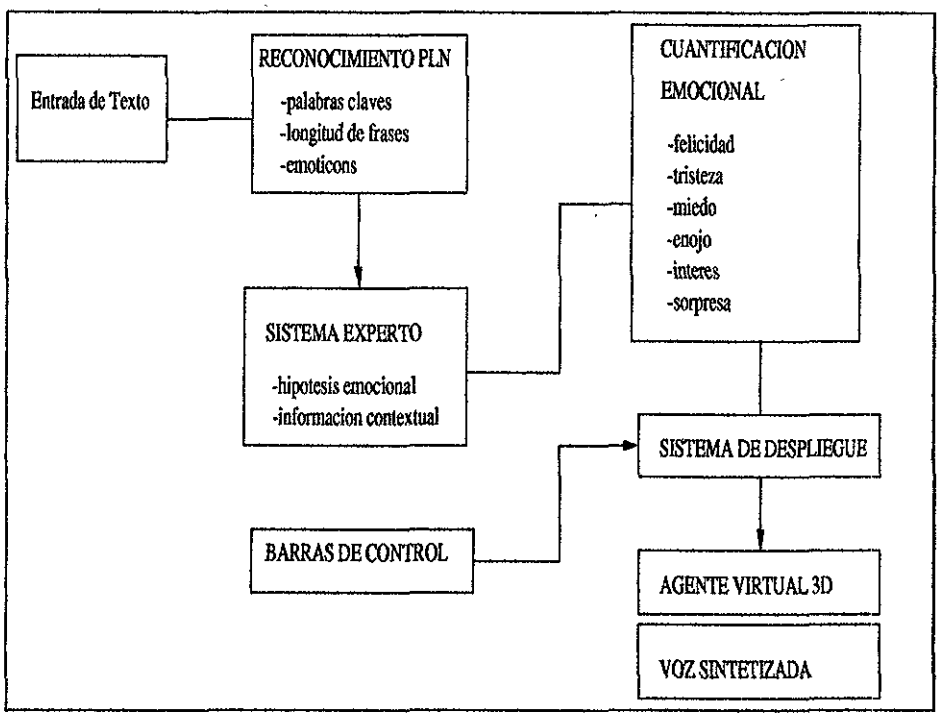

Figura 8.1: **Diagrama a Bloques del Sistema.**

De igual manera se tiene contemplado que en caso que el sistema reconozca una emoción diferente de la que se quería desplegar, exista una ventana con barras de movimiento en la cual se pueda modificar esta emoción en forma manual.

Por otro lado de acuerdo al diagrama mostrado, existen dos tipos de módulos principales que utilizaron diferentes plataformas de implementación: los módulos de reconocimiento y cuantificación de las emociones, y los módulos de visualización y animación. Estos módulos se acoplaron através de programas de comunicación con la red.

Este sistema se implantó dentro de un *Chat* donde existen 2 o más personas manteniendo una conversación entre sí. Para la comunicación entre los usuarios se utilizan rutinas de Sockets desarrolladas en C las cuales también comunican los dos módulos principales sin importar la plataforma de desarrollo, pues estas rutinas funcionan através de la red.

8.3.1 Representación de Emociones en forma textual.

En el diálogo sostenido cara-a-cara los humanos usan un amplio rango de señales no verbales para reconocer las diferentes emociones por ejemplo: felicidad, tristeza, enojo. Estas señales estan ausentes en entradas solo-texto, sin embargo pueden encontrarse otros signos utilizados que señalan la presencia de estos estados emocionales. Estas señales pueden ser explícitas (como lo muestran las 3 primeros descripciones) o implícitas (como el resto de la lista) incluyendo:

1. Tipos de palabras utilizadas, como adjetivos describiendo estados de ánimo por ejemplo: feliz, triste, enojado.
2. Uso de emoticons, como :-) or :-D muy utilizados en diálogos textuales para mostrar felicidad o tristeza.
3. Uso de adverbios.
4. Uso de letras mayusculas para denotar enojo.
5. Longitud del enunciado, por ejemplo: cuando una persona se encuentra enojada la longitud usual de estas frases es breve, claro esta que esto viene a ser parte de una comprobación del estado emocional que se reconoció.
6. Número de errores textuales, lo cual sucede cuando una persona se encuentra en una emoción muy intensa.

7. Información contextual.

Algunas de estas señales son dificiles de detectar y en este trabajo solo nos ocupamos de las señales explicitas. En el presente trabajo se utilizan los puntos 1,2 y 3, y se tiene la posibilidad de aumentar los demás puntos restantes.

En los últimos años los sistemas de comunicación por computadora han dado lugar a nuevas formas de comunicación emocional las cuales tratan de compensar la falta de elementos no verbales como son los gestos y despliegues faciales [KERS97].

Muy comúnmente estos sistemas y en especial los Sistema de Platicas Remotas (SPR) utilizan representaciones especiales, denominadas *emoticons*, que infieren ciertos estados de ánimo. Éstas representaciones se visualizan como caracteres especiales por ejemplo dos puntos junto a un paréntesis que cierra para mostrar una carita representando alegría.

Kershaw [KERS97] menciona que estos emoticons tal vez son lingüísticamente innecesarios, pero debido a su difusión se han convertido en una forma de identificación dentro de un grupo social. Sin embargo también muestra que muchos emoticons sí tiene la labor de mostrar estados de ánimo que de otra manera serían difíciles de distinguir. La siguiente tabla muestra algunos de los emoticons más utilizados en los SPR para expresar emociones. Estos son tomados del libro de Damer [DAME98].

Caracteres de Emoticons	**Significado**
1. :-)	feliz
2. :-D	muy feliz
3. :-(	triste
4. ;-)	pestañeo
5. (:-(	muy triste
6. :-O	impresionado
7. :-|	sin reacción

Tabla 8.1. Modelo Emocional para el Despliegue Facial.

Hasta el momento se reconocen los *emoticons* 1,2 y 3 los cuales afectan las categorías de felicidad y tristeza.

De igual manera se hace reconocimiento de signos de admiración para dar mayor énfasis a las emociones, al realizar la conversación.

8.3.2 Módulos de Reconocimiento y Cuantificación de Emociones

Estos módulos pertenecen al sistema experto a cargo del reconocimiento, razonamiento y evaluación de la emoción correspondiente una vez que se adquirió una entrada de texto. El sistema experto correspondiente fue desarrollado usando también CLIPS [GIAR94].

Cuantificación de Emociones y Lógica Difusa

Para obtener los valores de intensidad se utilizó un proceso de cuantificación con cambios continuos. Por lo que para resolver este problema se utilizaron dos funciones combinadas para obtener el grado deseado de la emoción presente. Una función exponencial y una transición que utiliza técnicas de lógica difusa.

Como se mencionó el proceso de cuantificación de emociones (capítulo 5) es asociado generalmente con una función exponencial negativa. De manera que si se cambia la intensidad de un estado al otro (por ejemplo de felicidad a enojo) la primera transición debe de presentar un cambio considerable. Y al continuar reforzando la emoción sobre el mismo rubro en este momento el valor se incrementará en forma exponencial, como se describe en la siguiente ecuación:

$$em_j = \sum_{j=0}^{N} em_j * e^{-i_j} \qquad \forall j \geq 0, i \geq 0 \tag{8.1}$$

donde:
em_j = valor de intensidad nuevo de cada emoción j
N = número de emociones que se van a modificar
i_j = valor asignado a la exponencial que produce cierto cambio en la intensidad de cada emoción

Tan pronto como el sistema reconoce una emoción nueva inicializa el valor de i a 0 y comienza de nuevo la cuantificación. Este método da un valor real que es usado como una entrada al siguiente paso: el sistema difuso.

Se escogió utilizar lógica difusa en la forma de razonamiento aproximado, porque es ampliamente usado en áreas que necesitan de un soporte para la toma de decisiones y un sistema experto con capacidades de razonamiento poderoso, así como el menor número de reglas posibles. Para un sistema difuso típico se tiene una entrada

(en este caso el resultado de la función exponencial) que pasa por el proceso de fuzificación y la salida la cual es defuzificada y envia los datos al sistema de control (en este caso hacia el módulo de control de despliegue facial de las expresiones). A continuación se explica con mayor detenimiento el proceso empleado.

Primero se definieron conjuntos difusos a utilizar por el sistema, otorgandoles valores de pertenencia o membresía a cada emoción dentro de un cierto rango de valores. Obsérvese en la Figura 8.2 como es que estos conjuntos difusos se crean para las emociones. En este sólo se muestran los estados pertenecientes a una sola categoría felicidad.

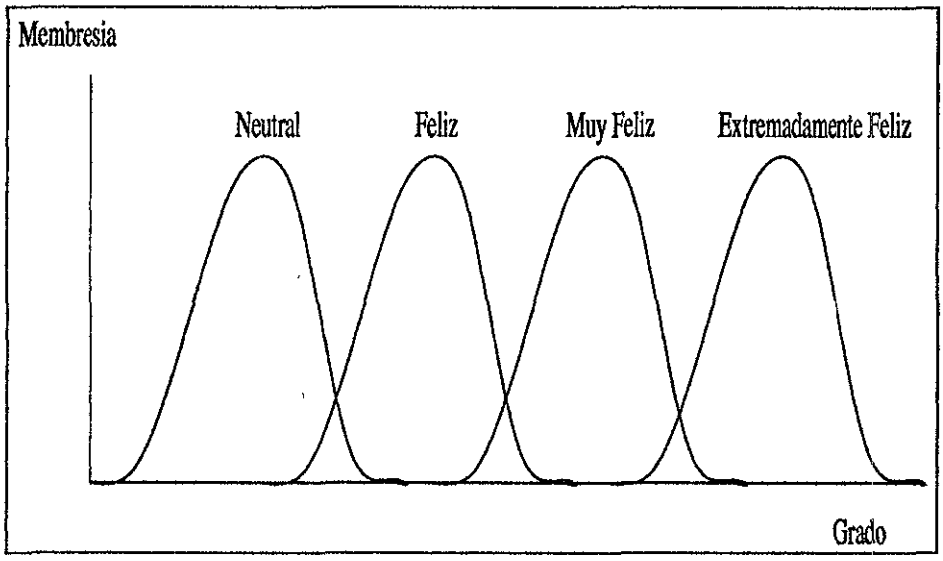

Figura 8.2: **Conjuntos difusos para la emoción Felicidad.**

Observe que estos conjuntos integran en su definición variables lingüísticas que también cuyo valor es afectado directamente por entradas textuales tales como: *muy, más, poco, extremadamente.*

Después de aplicar las variables lingüísticas que modifiquen el valor de la emoción dentro de los conjuntos difusos, se procede a trabajar con los grados de verdad que le corresponden al valor asignado con estos valores μ (o grado de pertenencia). El grado de pertenencia afectará en cierto porcentaje a cada conjunto difuso. Puede suceder que el valor obtenido sólo afecte a un conjunto y entonces únicamente afecta a este conjunto en su totalidad. Pero si no es así se realiza una suma lógica ó suma de áreas de los conjuntos difusos que contribuyen en cierto grado a la membresía.

Ver capítulo 5 para mayor detalle.

Y acto seguido se inicia una etapa encargada de obtener un valor representativo de estas contribuciones o áreas de los conjuntos difusos. Etapa conocida como defuzificación. El proceso de defuzificación utilizado fué el algoritmo del centro de gravedad o centroide. La función del defuzificador es el encontrar la mejor posición dentro del conjunto difuso. Lo cual implica generalmente que los algoritmos de defuzificación son un compromiso entre la necesidad de encontrar un punto resultante y la pérdida de información por tal proceso.

El algoritmo del centro de gravedad encuentra el punto de balanza dentro de la región difusa calculando la media de los pesos de las regiones. Aritméticamente la solución dentro de la región A del conjunto difuso en el caso discreto se define por:

$$cen = \frac{\sum_{i=0}^{n} d(i) \times \mu_A(di)}{\sum_{i=0}^{n} \mu_A(di)} \tag{8.2}$$

Al aplicar este algoritmo se obtiene una salida númerica mas precisa y lista para usarse por la etapa de despliegues emocionales. Toda la etapa del proceso de lógica difusa y asignación de valores emocionales a partir de un texto se realizó con el sistema experto CLIPS.

Manejo de Emociones en CLIPS

La manera en que CLIPS maneja las emociones es de igual forma que los guiónes, a través de objetos. En la tabla 8.2 se puede observar el campo que contiene la información de las palabras o sinónimos que la activan. Así como el campo con la emoción que modifica su intensidad en forma negativa. Otro de los campos enseña el grado de intensidad que toman estas emociones.

Emoción "Felicidad"	
identificador	felicidad
grado	neutral, feliz, más feliz, extremadamente feliz
operando	verdadero o falso
sinonimos accionantes	bien, contendo, excitado, simpatico

Tabla 8.2. Atributos de un objeto emocional.

Existe un objeto definido para cada emoción. Y para el reconocimiento se creó otro que lleva toda la información sobre las intensidades que se poseen para cada emoción, hasta el momento. Vea la Tabla 8.3.

Registro de emociones	
felicidad	intensidad
tristeza	intensidad
enojo	intensidad
sorpresa	intensidad
miedo	intensidad
interes	intensidad
operando	falso o verdadero

Tabla 8.3. Ejemplo de un objeto para un situación.

Obsérvese que cada emoción guarda un campo para el valor final de la intensidad. Finalmente en base a estas intensidades obtenidas, al proceso de lógica difusa y por último a la desición del sistema experto sobre la emoción que debe ser desplegada se obtiene el resultado.

El sistema experto también se encarga de convertir la información a un formato adecuado para los siguientes módulos que se explican a continuación.

8.3.3 Módulos de Animación y Visualización

Este módulo es el despliegue facial que utiliza animación muscular basada en interpolación. Este despliegue es el DECface desarrollado en los laboratorios de investigación de la compañía *Digital Equipment Corporation* [WATE93]. Para lograr la animación se modifico el código de software inicial, de manera que puediese realizar animación atraves del metodo de interpolación que se explica en el capítulo 6. Y para completar esto se utilizó el Sistema Codificado de Acciones Faciales SCAF (capítulo 6), el cual describe movimientos faciales a través de acciones musculares.

Para la definición de estos movimiento se utilizaron los músculos de la tabla 8.4. Estos se tomaron del sistema SCAF e intervienen en el modelo de este trabajo.

Movimiento muscular	*Rango de valores*
I_Zygomatico_Mayor	-1 a 1
D_Zygomatico_Mayor	-1 a 1
I_Angular_Depresor	-1 a 1
D_Angular_Depresor	-1 a 1
Izq_Frontal_Interior	-1 a 1
Der_Frontal_Interior	-1 a 1
Izq_Frontal_Mayor	-1 a 1
Der_Frontal_Mayor	-1 a 1
Izq_Frontal_Exterior	-1 a 1
Der_Frontal_Exterior	-1 a 1
Izq_Labio_Nasal	-1 a 1
Der_Labio_Nasal	-1 a 1
Izq_Interior_Labio$_{N}$*asal*	-1 a 1
Der_Interior_Labio$_{N}$*asal*	-1 a 1
Izq_Lateral_Corrigador	-1 a 1
Der_Lateral_Corrigador	-1 a 1
Izq_Secundario_Frontal	-1 a 1
Der_Secundario_Frontal	-1 a 1

Tabla 8.4. Unidades músculares de movimiento.

"I" significa músculo de la parte izquierda de la cara, y "D" músculo de la parte derecha. Y en base a estas unidades musculares se contraen o aflojan los mismos. Se estableció un rango de parámetros que van de -1, para contraer hacia arriba o a la izquierda, hasta 1, contraer hacia abajo o a la derecha.

Este modelo es el descrito por Parke en el cual se tiene un secuenciador que checa el descriptor paramétrico del movimiento, el cual tiene la estructura:

$< nombre\_del\_musculo >< parametro\_inicial >< parametro\_final >$

donde el método para crear una secuencia entre el cuadro inicial y el cuadro final es el de interpolación.

También existen movimientos tales como el de rotación en los cuales se trata el despliegue como una imagen, así como uso del modelo de luz y sombras que posée OpenGl. Esto se logra aplicando las técnicas descritas en el capítulo 6.

En breve este módulo consiste de una cara que corresponde al agente que se desea desplegar y se encuentra modelada en 3 dimensiones, vease Figura 8.3. La versión

actual consiste de 500 polígonos. Y la cara es cubierta con una capa que simula la piel de manera que luzca real. Para simular la piel los polígonos se recubren con superficies que se suavizan de manera que parezca mas real. La transformación de una expresión a otra se realiza a traves de interpolación que provoca una deformación de los polígonos. El sistema almacena los valores máximos asignados a los músculos (del sistema SCAF) y para cada expresión, en un archivo que se carga al inicio de la ejecución del programa.

Para una mayor información sobre OpenGL refiérase al apéndice B, donde se muestran las operaciones básicas necesarias para crear imágenes constituídas de polígonos.

Una adición al proyecto es el uso de una salida especial con síntesis de voz a partir del texto escrito. La adición de afecto a la voz sintetizada es útil en cualquier aplicación en la cual la expresión es deseable dado que de esta manera se obtiene comunicación más natural [CAHN89]. Este módulo puede simular una voz de felicidad, tristeza, enojo, miedo y normal de una persona, usando voces femenina, masculina y de niño según se seleccione. El sistema experto también se encarga de coordinar la voz sintetizada.

Como se mencionó uno de los aspectos importantes de la interface será retroalimentación visual, es decir, el usuario sera capaz de ver como luce su avatar y si es necesario, cambiar su expresión manualmente. De esta manera idealmente se asegura que el avatar posea siempre la expresión que el usuario desea. Sin embargo se espera que los módulos del sistema experto controlen esto en forma inteligente y eviten esta situación. La representación visual fue desarrollada en el lenguaje de programación **OpenGL**. Y el sistema manual de control utiliza **Motif**, que es un lenguaje de macros escrito en lenguaje C. El módulo de entrada de los usuarios consiste de barras que permiten un movimiento manual de los diferentes músculos faciales.

Este despliegue utilizado aún se encuentra en pruebas preliminares pues también se han contemplado otros despliegues que pudiesen poseer mayores ventajas respecto al utilizado por el momento.

Las figuras 8.4 y 8.5 muestran resultados del avatar con la expresión de felicidad y la expresón de enojo.

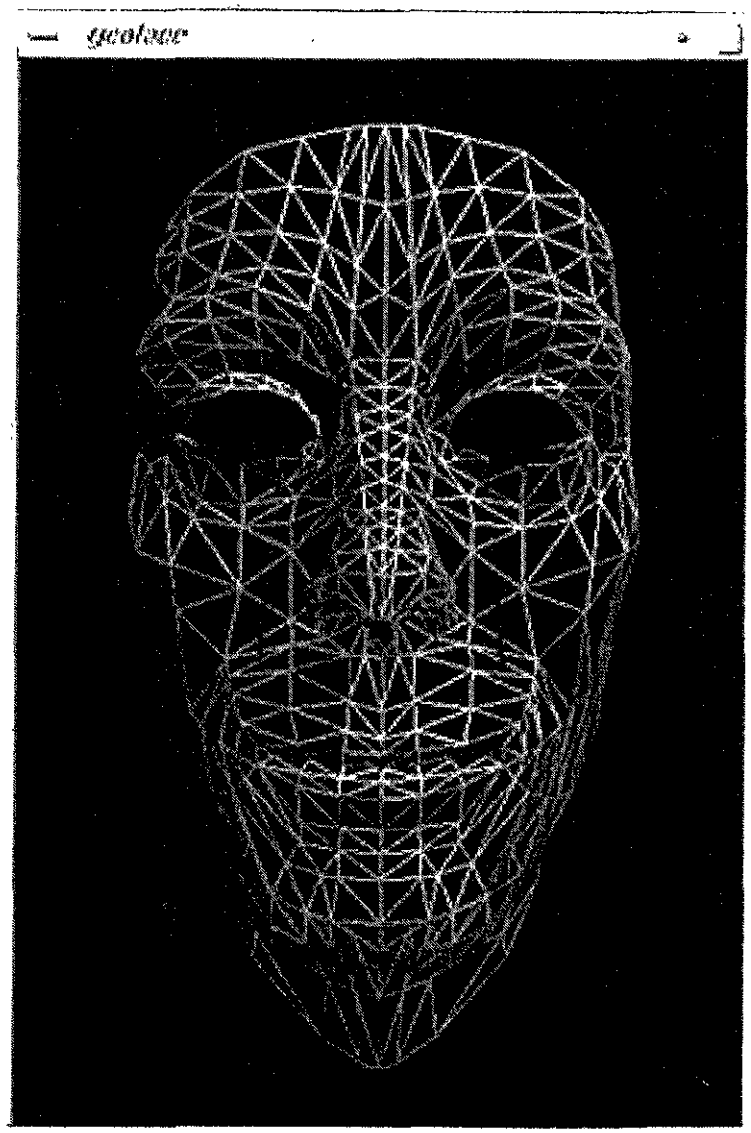

Figura 8.3: **Polígonos constituyentes del despliegue facial.**

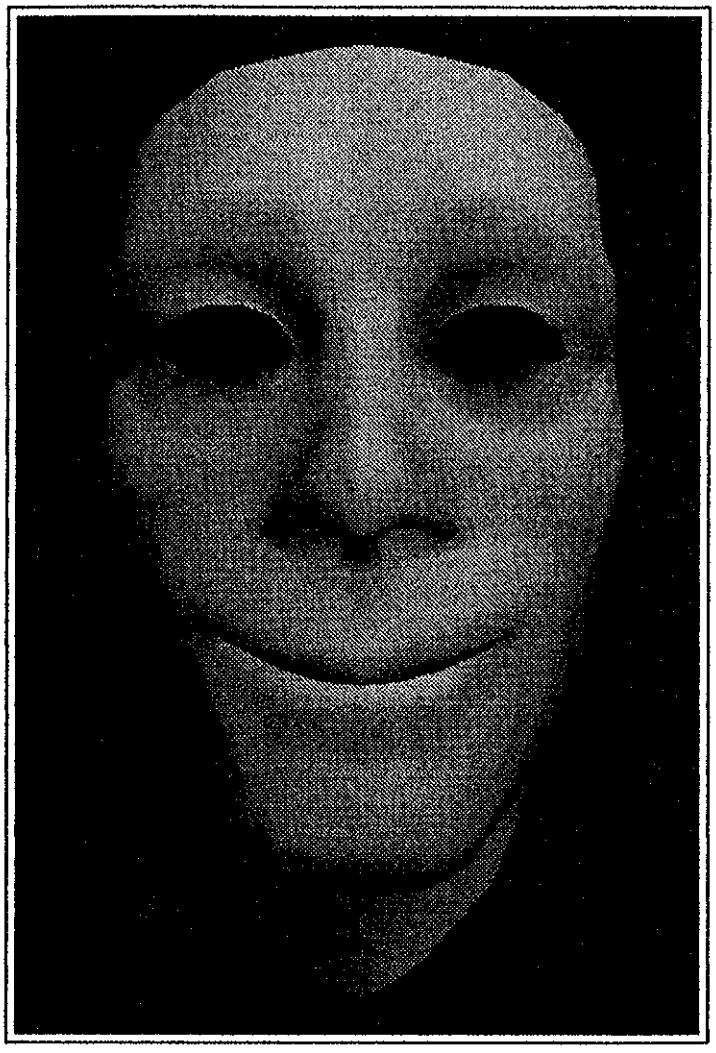

Figura 8.4: **Despliegue Facial para Felicidad.**

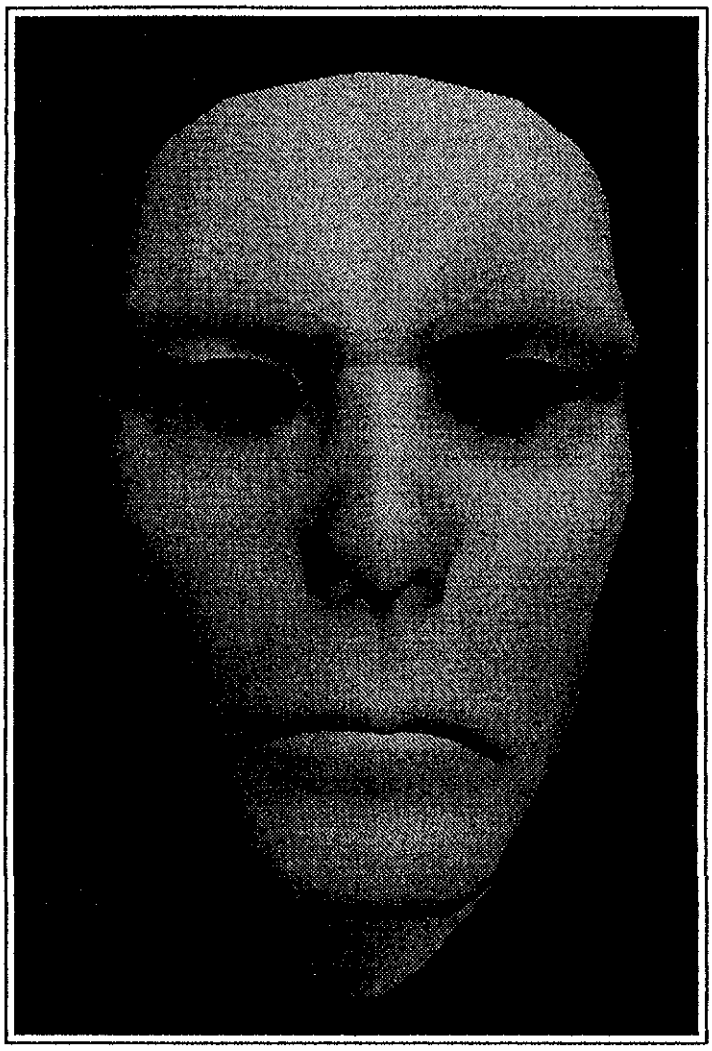

Figura 8.5: **Despliegue Facial para Enojo.**

Capítulo 9

Resultados

9.1 Agente con Reconocimiento de Guiones

A continuación se muestra en la figura 9.1 el ambiente gráfico que se utilizó para observar las pruebas en el robot.. Aunque cabe mencionar que este sistema se desarrolló para ser utilizado en un robot real.

9.1.1 Resultados Experimentales

Se crearon tres guiones durante la etapa de entrenamiento: los cuales corresponden a los guiones de bañar al perro, el guión de hacer la comida y el guión de recoger los objetos tirados en la casa. Estos guiones poseen una complejidad mayor a la del guión de muestra presentado en el capítulo 5. Contienen al menos 6 etapas a cumplir más todas aquellas instrucciones primitivas que derivan de sus pasos básicos.

En primera instancia se comprobó que los guiones almacenen correctamente las etapas constituyentes, así como asegurarse de que se puedan guardar en un archivo de texto. Así cada vez que se utilice el programa, los guiones se cargaron automáticamente como parte de la base de datos del sistema encontrándose listos para reconocerse. De igual manera si el usuario desea añadir los guiones sin tener que ordenarlo al robot, se puede escribir a este archivo, teniendo el conocimiento previo sobre como se decodificaron las órdenes en primitivas.

Una vez creados se procedió a probar su reconocimiento ordenando comandos relacionados con el guión. Es decir se ordenan uno a la vez y se comprobó que el guión de reconocimiento fuera almacenando estas órdenes y al mismo tiempo comparando contra los guiónes pre-existentes en todo momento. Una adición importante

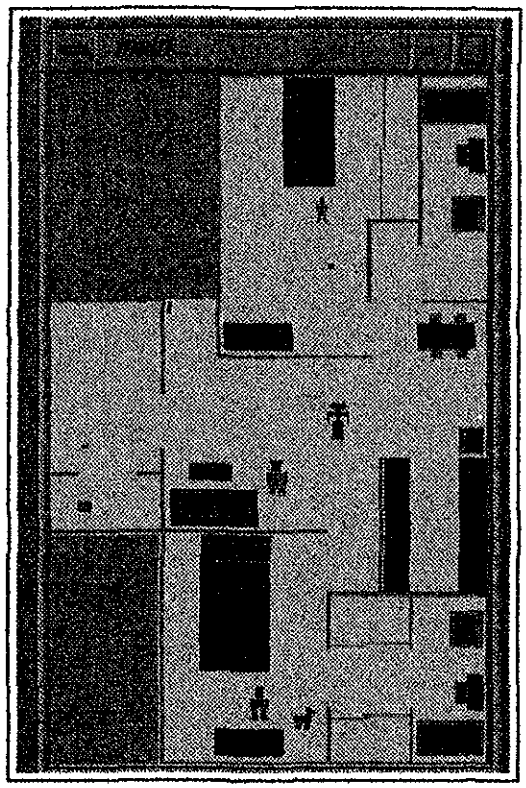

Figura 9.1: **Gráficos utilizados para probar el funcionamiento de los Guiones.**

a considerar, fue que en el momento en que el robot se encuentra en un cuarto donde se ejecutan frecuentemente ciertos guiones, como puede ser el preparar la comida en la cocina, provoca que se habilite este guión en especial para su reconocimiento.

Al ir ordenando comandos relacionados con un guión, al momento que se encontró una similitud del 50% se disparó el guión reconocido y ahora sí las ordenes restantes se aislaron y decodificaron nuevamente en dependencias conceptuales, ejecutándose en el orden prescrito por el guión.

9.2 Avatar Emocional

En la figura 9.2 se muestra el sistema completo en funcionamiento y la interface que ve el usuario. Observe que se tienen los siguientes componentes del sistema: una interfaz gráfica de la persona que se comunica, una interfaz de la otra persona con

quien se comunica, las barras de control manual, dos ventanas de mensajes (una de escritura y otra lectura de mensajes), y por último el sistema experto que obtiene la emoción presente.

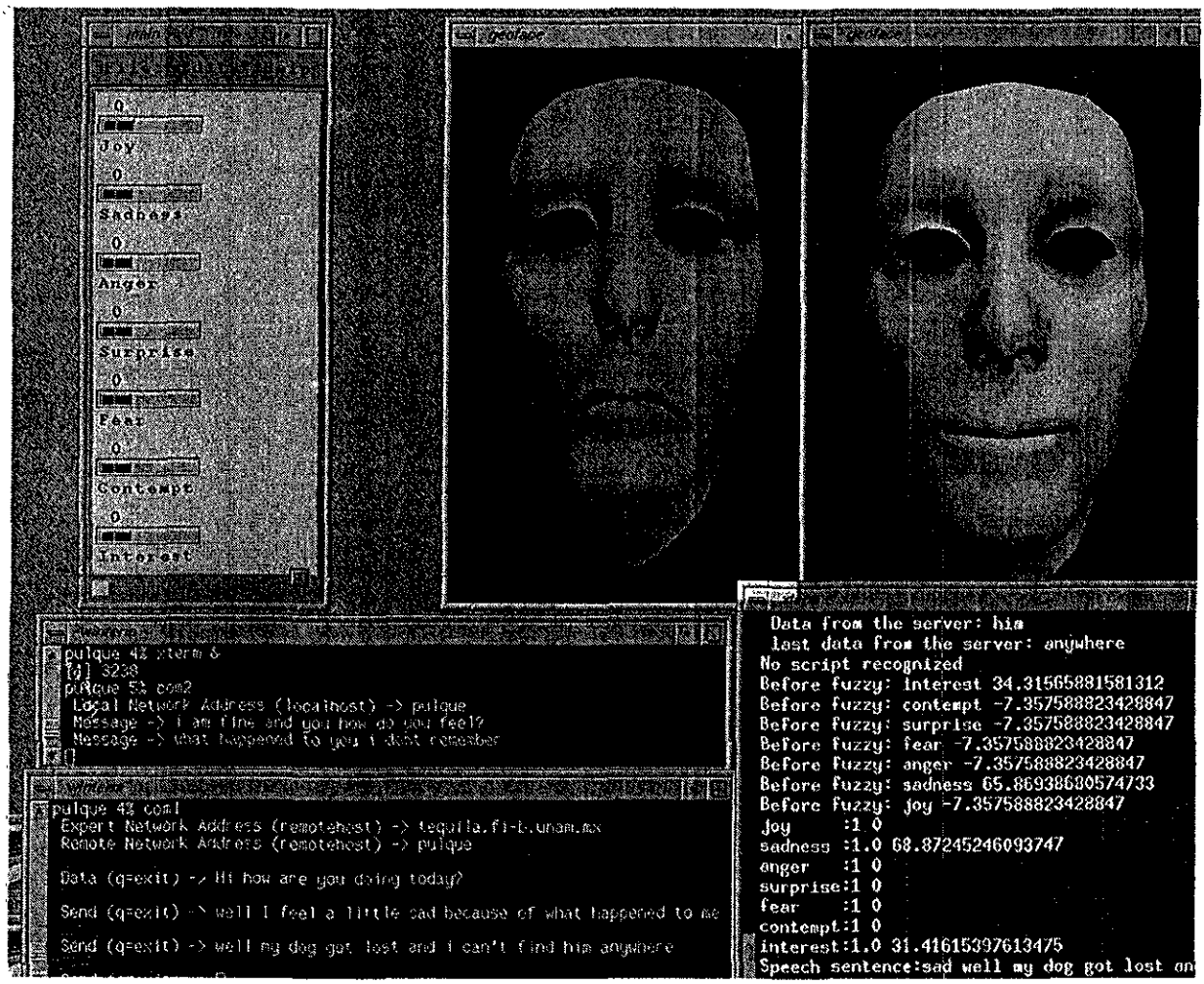

Figura 9.2: **Sistema Completo que observa el usuario.**

9.2.1 Resultados con Usuarios en el HITLab

Para evaluar el efecto al añadir despliegues emocionales a una interface de SPR se condujo un estudio sencillo. Este estudio se realizó en el Human Interface Technology Laboratory (HITLab) de la Universidad de Washington, Seattle. Participaron veinte sujetos estableciendo una comunicación en inglés, por lo que se formaron 10 parejas. Sus edades oscilaron entre 18 a 34 años y todas habían tenido experiencia previa con las computadoras, al menos vía email. Para comprobar las ventajas con respecto a un SPR normal, se comparó la comunicación usando 3 diferentes condiciones en el sistema que se muestran en la tabla 9.1. Estas condiciones de prueba no siempre se hicieron en el mismo orden para evitar que este factor pudiese influir en los resultados.

Condiciones de Prueba	Tiempo [min]
Sólo texto	15
Texto con representación del Avatar Estático	15
Texto con representación del Avatar Emotivo	20

Tabla 9.1. Condiciones y tiempo de pruebas del Sistema.

En la condición de sólo texto los usuarios manejaban una ventana de texto mediante la cual se comunicaban y contenían sus comentarios. La segunda condición añade el despliegue de un avatar sin movimiento, mientras que la tercera utiliza el sistema experto para mostrar los despliegues emocionales del avatar.

Antes de que los sujetos comenzarán a comunicarse se les preguntó las palabras que comúnmente utilizan en los emails para expresar emociones. Esto fue con el propósito de aumentar el vocabulario del sistema. También de que los participantes comenzaran con el tópico de discusión se les dió cinco minutos para que se hicieran preguntas generales y aprendieran el funcionamiento del sistema. Esto aseguró que los usuarios observaran los 7 estados emocionales con los que se cuenta.

Como resultado de las pruebas se identificaron 37 nuevas palabras que representan emociones. Estas palabras se añadieron al sistema así como se quitaron otras, quedando un total de 95 palabras.

Acto seguido se les pidió que escogieran entre diversos temas de conversación: cine, autos, trabajo, la última gran fiesta que recordaran, la mejor aventura que

tuvieron, el mejor viaje, una situación tenebrosa, animales o como sugerencia, relatar un cuento fantástico. Las condiciones sólo texto y texto con representación del avatar estático tuvieron una duración de aproximadamente de 15 minutos cada uno, y la tercera fue de 20 minutos. Para medir estas condiciones y obtener aspectos importantes para nosotros se les pidió contestaran un cuestionario.

También se les solicitó a los sujetos asociar los siete estados emocionales con las expresiones faciales que se poseen. Inicialmente se mostraron las caras pidiendo su reconocimiento. También se mostraron los despliegues en orden diferente para cada sujeto de prueba. A continuación se mostraban etiquetas para que hacerlas coincidir con las caras. La tabla siguiente muestra los porcentajes de reconocimiento de las caras con respecto a cada emoción:

Despliegue Facial	**Reconocimiento [%]**
felicidad	100
tristeza	95.23
enojo	95.23
sorpresa	14.28
miedo	38.09
disgusto	66.66
interés	38.09

Tabla 9.2. Porcentajes de reconocimiento de Despliegues Faciales.

Lo anterior demuestra que la expresión facial de **sorpresa** es la que presenta m'as problemas para reconocerse. Y **felicidad, tristeza** y son las más fácilmente reconocidas. Los despliegues emocionales **sorpresa, interés** y **miedo** fueron los que causaron mayor confusión. Tambien se realizaron preguntas que calificaran que tan real era la expresión en una escala del 1 al 10 siendo las menores calificaciones para los despliegues de **sorpresa, interés** y **miedo**, lo cual corroboró los porcentajes que se habían obtenido de la prueba anterior. Al parecer es dificil distinguir **interés** de **felicidad**, asi como **miedo y tristeza**. Otros despliegues que tambien llegaron a confundirse entre sí fueron **enojo y disgusto**, que no consideramos tan importante pues son emociones que poseen cierta semejanza.

De lo anterior se obtuvo que un 90 % de los participantes no observaban ninguna diferencia significativa al realizar las condiciones de comunicación solo texto y texto con representación del Avatar Estático, salvo por el principio al observar la imagen,

ya que los sujetos tendían a perder interés en la representación gráfica concentrándose en la conversación. Los porcentajes de los resultados obtenidos de acuerdo a su funcionamiento y agrado general del sistema se muestran en la tabla 9.3.

Comportamiento de usuarios	Porcentaje
Usuarios que les gusto	75%
Usuarios que piensan es demasiado complicado	15%
Usuarios que no lo volverían a usar	10%

Tabla 9.3. Resultados en porcentajes según opinión de los usuarios hacia el sistema.

Al utilizar el Avatar Emotivo se observó que los usuarios que manipularon con mayor facilidad el sistema, tendieron a poner atención a las caras con mayor frecuencia y trataron de accionar las diferentes emociones tanto en el reconocimiento de palabras como en las barras de movimiento. El 85% por ciento de los sujetos acertaron en que les parecía una manera mas eficiente de charlar haciendo mas natural la comunicación. Esto indica que la presencia de avatares realmente tiene un impacto en la conversación.

Aparte de las palabras que denotan emoción y que se les preguntaron a los usuarios también se logró ampliar el vocabulario real debido al monitoreo constante de las conversaciones, lo cual ayudó a localizar nuevas palabras que mejorasen el sistema.

En cuanto a las pruebas hechas con las tres condiciones se obtuvo que los usuarios no observaron una gran diferencia entre las condiciones solo texto y texto con avatar estático. Esto fué porque sólo notaban la presencia del avatar al inicio de la conversación, pero después tendían a olvidarse del mismo y concentrarse únicamente en la conversación. Otro de los resultados observados fue el hecho de que los usuarios prefirieron cambios más drásticos los cuales se lograban mejor sin la parte de lógica difusa al sistema. Es decir sin cambios mas naturales, sino más exagerados.

También surgieron posibles modificaciones de estética al sistema que se piensan tomar en cuenta para poder lograr una visión mas natural del despliegue. Tales sugerencias fueron el agregar cabello, ojos y su movimiento. De igual manera se sugirió el unir las ventanas de despliegue facial con la de las respuestas textuales. Y así arreglar cualquier problema de sincronía entre las respuestas que llegaban a

surgir y los despliegues faciales, pues aunque solo se retrasaban por 1 segundo ésto parece ser un punto clave para el usuario. De igual forma hubo el comentario de que faltaba un despliegue de interrogación. Razón por la cual se añadió esta última al momento de detectar una pregunta junto con las palabras como, que, cuando y donde.

Capítulo 10

Conclusiones

Tradicionalmente los caracteres gráficos de computadora son completamente autónomos o enteramente controlados. Sin embargo existe una tercera clase de caracteres que poseen un cierto nivel de autonomía que permiten al usuario controlar en cierta medida la comunicación humano-computadora. Como puede verse el objetivo de esta tesis es ofrecer herramientas que faciliten el manejo de estos caracteres y amplien la comunicación. A continuación se presentan las principales conclusiones obtenidas como resultado de este trabajo.

Una justificación importante de esta tesis es el de haber comprobado la necesidad de utilizar en ambas aplicaciones un sistema de control experto. La capacidad de decisión de los sistemas expertos y el manejo de información que tienen, otorgaron una organización adecuada en base al estado interno del sistema, lo que los hace más eficientes al interactuar con su entorno. Esto se observó al mostrar la facilidad de representar a cada actor, cosa, lugar y emoción como un objeto y tener un manejo sencillo que le ayudó a modificar su estado. En base a su conocimiento, constituído de la observación al crearse nuevos hechos y su conjunto de reglas de producción el sistema tuvo la habilidad de desempeñarse sin ningún estímulo externo. Lo cual muestra que los sistemas expertos son una manera efectiva de codificar conocimiento experto para la resolución de problemas cuyo dominio es específico.

En el caso de inferencia de emociones se observó que existen no solo los hechos mencionados a medir, sino también se puede crear un sistema tan sofisticado como se desee al ir agregando más conocimientos y observaciones generales durante su funcionamiento. Un ejemplo puede ser el añadir el manejo de mayúsculas para inferir si la persona esta gritando. También se tiene otro caso cuando los usuarios utilizan la barra de control de emociones con ciertas expresiones. Esto último representa

una posible extensión que otorgue "aprendizaje autónomo" al sistema experto.

Por otro lado el hecho de agregar que los guiones sean sensibles al contexto al ejecutar el guión correspondiente en ese momento hizo que se maneje cierta incertidumbre, deseada en todo sistema experto.

Una extensión futura deseable en ambos sistemas expertos es el que también manejen la toma de decisiones en base a la experiencia propia adquirida en su entorno lo cual le otorgaría la característica faltante para ser un sistema con posibilidad de aprendizaje autónomo. Se propone un mecanismo basado en probabilidades preestablecidas, las cuales puedan ir modificándose y por lo tanto modifiquen su conocimiento.

Agentes Autónomos

Uno de los objetivos importantes de esta tesis fue el demostrar la capacidad de los agentes para interactuar en diversos proyectos explorando la utilización de algoritmos y prototipos de sistemas. Como se muestra en las aplicaciones que se realizaron los agentes de computadoras habitan y modifican su ambiente dinámicamente, ya sea dentro de una casa o mostrando respuestas emocionales ante un usuario. Al realizar estos sistemas se tomó en cuenta las metas presentes que poseen los agentes y como logran llegar a estas metas.

En el caso de los guiones se basaron en experiencias pasadas así como en sus entradas sensoriales para la construcción de primitivas que definieron el comportamiento del agente. De esta manera el agente obtuvo tres de las características más importantes para considerarse como tal: reactividad, al ser capaz de reaccionar ante las tareas que se le pedían; proactividad, al tomar iniciativas de encontrar maneras de ayudar a los usuarios; y aprendió como adaptarse al ayudar en tareas repetitivas

Para las respuestas emocionales de igual manera se puede decir que cumplieron con las características de agentes como son: reactividad, un cierto conocimiento del lenguaje así como una capacidad de inferencia que hace emerger los despliegues; continuidad en su ejecución; y personalidad al mostrar emociones.

Se concluye de todo lo anterior que la diferencia entre agentes de software y otro tipo de programas es conjuntar la habilidad de trabajar en ambientes distribuidos de computación y la habilidad de mostrar un conocimiento para automatizar las tareas de los usuarios al cooperar y coleccionar datos del mismo de manera personalizada. El problema que se deja abierto es el establecer los mecanismos de cooperación de

estos agentes con otros agentes.

Los siguientes parráfos son conclusiones más particulares a cada aplicación.

Sistema de reconocimiento de Situaciones

El objetivo primero de este sistema fue comprobar el funcionamiento de la creación y reconocimiento de situaciones cotidianas, las cuales pueden representarse por medio de guiónes. Mediante estos guiónes se mostraron características generales aplicables a cualquier otro tipo de sistema es decir, no únicamente al entorno de un robot dentro de una casa.

Es importante notar que la capacidad de aprendizaje de este sistema dependió del grado de inferencia que se tuvo sobre los hechos a su alrededor y la manera de aplicar este conocimiento en los demás aspectos cotidianos. Al utilizar este tipo de razonamientos en el caso del robot se logró una emulación de como generar el conocimiento, aunque se debe aclarar que este proceso es mucho más complejo.

El uso de dependencias conceptuales le otorgó al sistema la característica de entendimiento de situaciones y como utilizarlo para resolver problemas. Estas estructuras de dependencias conceptuales son sencillas de utilizar debido a que información diferente (en este caso sentencias gramaticales) con el mismo significado se representan por una misma estructura de datos al usar las mismas primitivas.

Por otro lado existe un acercamiento al problema de inteligencia artificial distribuida utilizando el sistema cliente servidor para comunicar las ordenes y ejecutarlas. Esto le otorga la capacidad de migrar hacia el internet. Además de que el hecho de manejar los demas objetos de la casa independientemente le permite descentralizar el almacenamiento de estos objetos y permitir que se compartan y modifiquen no solo por un agente sino varios. Sin embargo aun falta añadir a los demás agentes de la casa como entes autonómos que actúen de manera independiente y que posean una visión de todos los cambios que ejecuten los demás agentes dentro de la casa. Es decir, tomar en cuenta las tareas que el agente hace y tareas que otros agentes hacen.

El sistema de reconocimiento de emociones

Se describió un prototipo de una avatar inteligente que infiere emociones de una entrada textual la cual automáticamente modifica la expresión facial del mismo. Usamos un modelo simple para estados emocionales posibles y un modelo exponencial con transiciones modificadas por métodos de lógica difusa.

Los experimentos realizados con usuarios mostraron que un 75% de ellos responden positivamente ante estos avatares emotivos dentro de una ambiente de pláticas remotas, un 15% lo encontraron complicado de manejar y un 10% no lo volverían a utilizar. Suponemos que con futuras mejoras estos porcentajes de personas que no les gusto disminuiran. Pero uno de los hechos significativos que se encontraron fue que la naturaleza de la conversación es modificada al tener respuestas emocinales en forma visual. Esto fue por el hecho de tener un despliegue facial emocional que influía para utilizar expresiones textuales más exageradas y con mayor frecuencia. Sin embargo se debe de realizar un mayor trabajo en la interfaz para que esta tenga una apariencia más natural y por supuesto, aumente la comunicación. Entre posibles arreglos se encuentra el tratar de reducir el tiempo de retardo entre la entrada de usuario y el cambio de la expresión, aunque tambien existen los tiempos de retardo de la red.

Para el futuro existen aún varias posibilidades a explotar utilizando esta interfaz. Dentro del área lingüística se encuentran: identificar que tipo de texto o palabras o símbolos son predictores precisos de emoción, desarrollar una base de datos mayor de adverbios o modificadores de emociones, reconocimiento de estructuras conversacionales de más alto nivel como frases. Y por el lado de la interfaz hacer adiciones como movimiento de boca, ojos, cabello y agregar de igual manera comportamiento más complejo como puede ser desarrollo de comportamiento de bajo nivel. Ejemplos de este comportamiento son: movimientos de cabeza al asentir o negar, parpadeo de ojos y cambios de la postura, giros a la derecha a la izquierda de la cabeza. De manera que se le otorgue una mayor autonomía.

Otra de las propuestas a futuro en la cual pueden aplicarse ambos desarrollos de esta tesis, es el uso de guiónes para la creación de respuestas que utilicen gestos combinados como cuando surge una sorpresa agradable, se utilizaría un guión de control. El cual ordene primero desplegar una sonrisa combinada con una mayor apertura de los ojos y al mismo tiempo mueva la cabeza hacia atrás para continuar con una sonrisa más amplia. Otro ejemplo es un guión que se active cuando no existan respuestas emocionales, ya sea moviendo de posición la cabeza y parpadeando los ojos cada 1 segundo como normalmente ocurre en una persona.

Apéndice A

CLIPS

CLIPS es un shell para realizar sistemas expertos que se empezó a desarrollar en el centro espacial Johnson de la NASA desde 1985 y que ya está en su versión 6.0.

CLIPS provee la base para desarrollo basado en reglas (característico de los sistemas de producción), orientado a objetos y a la programación por procedimientos. Como se mencionó la programación basada en reglas permite la representación del conocimiento en forma de heurísticas o secuencias de reglas que especifican un conjunto de acciones realizarse en una cierta situación.

La programación orientada a objetos de CLIPS se denomina CLIPS Lenguaje orientado a objetos (en ingles COOL) y es una combinación híbrida de diferentes características que poseen otros lenguajes como el Common lisp Object System y SmallTalk. Las características que posee COOL incluyen clases con herencia multiple, abastracción, encapsulamiento, polimorfismo, y manejo de mensajes. La programación orientada a objetos permite modelar a los sistemas complejos como componentes modulares (reusables por otros sistemas o crear nuevos componentes). Las características de programación procedural son similares a las que otorgan el lenguaje C, ADA y Pascal y es sintácticamente similar a LISP. Como se puede ver con CLIPS se desarrolla software utilizando solo programación basada en reglas, programación basado solo en objetos o solo programación por procedimientos o bien una combinación de las tres [GIAR94].

A.1 Interacción con CLIPS

CLIPS es un intérprete, así que los programas escritos para él se pueden alimentar mediante el teclado o a través de un archivo. En UNIX el comando para ejecutar el

intérprete es:

```
\%clips [-f <archivo>]
```

Si la opción -f no se alimenta, el programa entra en su modo interactivo y admite instrucciones del usuario. Si por otro lado la opción -f se introduce, el programa busca el `archivo` y lo ejecuta.

A.2 Elementos básicos de programación

CLIPS admite ocho tipos de datos: float, integer, symbol, string, external-address, fact-address, instance-name e instance-address. Ninguna variable tiene que declararse, ya que CLIPS asigna el tipo en la primera asignación recibida.

A.2.1 Representación de datos

El mecanismo que más se utiliza en CLIPS para representar la información son los hechos. Los hechos se almacenan en una lista que también se conoce como base de hechos y forman la unidad fundamental de datos utilizada en las reglas. Las operaciones que se les pueden aplicar son: agregar (`assert`), retirar (`retract`), modificar (`modify`) y duplicar (`duplicate`). Estas operaciones se realizan dentro de las reglas o a través de interacción directa con el usuario.

Un hecho puede tener dos formas: ordenado o no ordenado. Un hecho ordenado es un simbolo seguido de cero o más campos separados por blancos y delimitado por paréntesis como en el caso de:

```
(robot uno 1 1 10)
(objeto caja 1 1 1 2 2 2 2 1)
```

Un hecho no ordenado se forma mediante la sentencia `deftemplate` que permite asociar a un dato un conjunto de slots o campos que pueden tener algún valor, cada campo es otro hecho. Por ejemplo:

```
(robot (posicion 1 1) (orientacion 10) (color azul))
```

El orden de los slots no tiene importancia.

En CLIPS los conocimientos se representan principalmente mediante reglas del tipo *si condicionantes, entonces consecuentes.* Estas reglas se escriben

A.2.2 Representación de conocimientos

Hay dos formas de representar el conocimiento. Los conocimientos heurísticos se representan mediante reglas de producción y los conocimientos procedurales mediante funciones.

Las reglas de producción se definen mediante la construcción `defrule` que tiene la siguiente sintaxis:

```
(defrule <nombre_de_regla> [<comentario>]
   [<declaración>]  ; Propiedades de la regla
   <elemento_condicional>* ; Lado izquierdo de la regla
 =>
   <acción>*) ; Lado derecho de la regla
```

los campos rodeados por corchetes son opcionales. Para incluir un comentario se introduce un ";" y CLIPS toma el resto de la línea como comentario. Los eloementos condicionales y la acción pueden ser cero o más. Las acciones son una secuencia de llamadas a funciones, afirmaciones o eliminación de hechos. Cuando se cumplen todos los elementos condicionales se ejecutan en orden todas las acciones.

Las funciones se definen con la construcción `deffunction` cuya sintaxis se muestra a continuación:

```
(deffunction <nombre_de_función> [<comentario>]
   (<parámetro regular>* [<parámetro comodín>])
   <acción>*)
```

Un parámetro regular es una variable de un sólo campo que se reconoce por estar antecedida por un signo de interrogación ?, mientras el parámetro comodín es una variable multicampo que se reconoce por estar antecedida de un signo de pesos seguido de la interrogación $?.

El resultado de la función es el resultado de la última acción desarrollada en la misma.

CLIPS se puede encontrar en Internet en `http://www.fi-b.unam.mx/soft.html`.

A.3 Representación en CLIPS de un guión

El siguiente guión con notación en CLIPS pertenece a la escena de bañar el perro. ([script-showerdog]

```
(ident shower-the-dog)
(actor robot)
(tools dog soap)
(location mothers-bathroom)
(ready ready)
(phases
(ptrans (actor robot)(to dog's_place))
(grasp (object dog))
(ptrans (object dog)(to mothers-bathroom))
(ptrans (actor robot)(to soap's_place))
(grasp (object soap))
(grasp (actor robot)(to towel's_place))
(grasp (object towel))
(operate false))
```

Apéndice B

OpenGL

OpenGL es un sistema de bibliotecas en C independientes para crear gráficas en tercera dimensión creando la impresión de perspectiva. OpenGL fue desarrollado por Silicon Graphics, Inc. y se encuentra disponible para cualquier plataforma de cómputo así como para cualquier sistema operativo.

El propósito de OpenGL es crear objetos de dos y tres dimensiones en un buffer para después desplegarlo en una ventana. Este lenguaje especifica como es que un objeto se crea geométricamente, y parte del principio de que toda figura se puede conformar por poligonos más simples.

B.1 Conceptos básicos de OpenGL

La creación de estas imagenes se realiza en varios niveles. A nivel básico de la imagen se manejan vértices. Un vértice es un punto, que puede ser el punto final de una línea o las esquinas de un polígono. Los vértices pueden ser bidimensionales o tridimensionales.

En el siguiente nivel se tienen primitivas, las cuales consisten de un grupo de uno o más vértices. Por ejemplo un cuadrado descrito por cuatro vértices es una primitiva.

Los vértices se unen para formar primitivas y estas a su vez para crear el buffer del polígono, que se controla a través de una variedad de parámetros y órdenes. Por ejemplo, la aplicación puede especificar como una matriz tridimensional define coordenadas del objeto a trasladar en una superficie.

Además de dibujar puntos y lineas, OpenGL puede dibujar superficies así como aplicar efectos de luz y uso de texturas o mapas de bits. Un diagrama simplificado

de como es que OpenGL trabaja se presenta en la figura B.1. Dentro de esta figura se pueden ver los niveles necesarios para crear una imagen en la computadora.

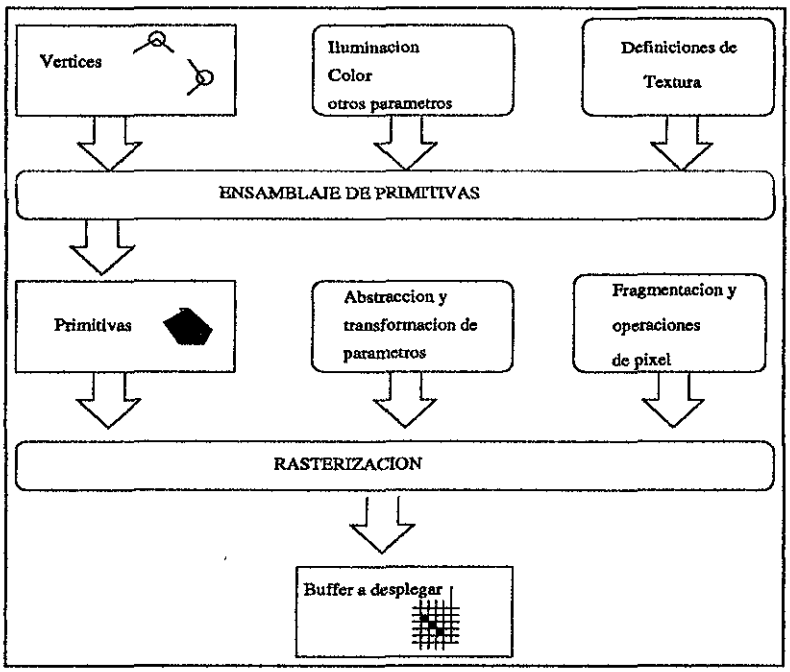

Figura B.1: **Operaciones de OpenGL simplificadas.**

El término rasterización se refiere a desplegar en pantalla la imagen, formada de pixeles.

B.2 Dibujando con OpenGl

La mayoría de las imagenes consisten de operaciones sobre vértices y se guardan entre las llamadas: *glBegin* y *glEnd*. La llamada *glBegin* identifica el tipo de primitiva cuyas operaciones subsecuentes se van a ejecutar en ciertos vértices; *glEnd* marca el fin de la construcción de la primitiva. Por ejemplo el siguiente programa construye un cuadrado negro en un fondo blanco.

```
void despliega(void)
{
   /* limpia todos los pixeles */
```

```
    glClear (GL_COLOR_BUFFER_BIT);
    /*establece el color del cuadrado negro */
    glColor3f (1.0, 1.0, 1.0);
    /*define el polígono y sus vértices */
    glBegin(GL_POLYGON);
    glVertex3f (0.25, 0.25, 0.0);
    glVertex3f (0.75, 0.25, 0.0);
    glVertex3f (0.75, 0.75, 0.0);
    glVertex3f (0.25, 0.75, 0.0);
    glEnd();
}
```

La función *glBegin* puede ser usada para definir una variedad de primitivas. La tabla B.1 enlista algunos de los parámetros permitidos para esta función.

Parámetro	Descripción
GL_POINTS	Crea una serie de puntos
GL_LINES	Una serie de líneas
GL_LINE_LOOP	Un grupo conectado de líneas
GL_TRIANGLES	Un conjunto de triángulos
GL_TRIANGLE_STRIP	Un conjunto de triángulos conectado
GL_QUADS	Un conjunto de cuadrilateros
GL_QUAD_STRIP	Un conjunto de cuadrilateros conectado
GL_QUAD_STRIP	Un polígono
GL_POLYGON	

Tabla B.1. Primitivas construidas con OpenGl.

Una vez definidas estas primitivas existen una serie de reglas que definen el comportamiento de estos vértices. Por ejemplo el vértice de un polígono se puede convertir de igual manera en el vértice de otro polígono.

B.3 Bibliotecas adicionales

Además de las operaciones básicas existen otras definidas por las bibliotecas OpenGL GLU (OpenGL Utility Library) las cuales contienen una serie de funciones que tratan

con el desarrollo de texturas, manejo de esferas, superficies y curvas. De igual manera poseen funciones para "romper" polígonos complejos o concavos en poligonos más simples.
OpenGL y OpenGLGLU se puede encontrar en Internet en `http://www.opengl.com`.

Bibliografía

[BRAD97] Bradshaw, Jeffrey et al. *Software Agents.* American Asociation of Artificial Intelligence Press. The MIT Press. 1997.

[CHOV91] Chovil, Nichole. *Social determinants of facial displays.* Journal of Nonverbal Behaviour. 1991 vol.15(3). Human Sciences Press Inc.

[CAHN89] Cahn, Janet E., *Generating Expression in Synthesized Speech.* Master's Thesis, Massachusetts Institute of Technology. Mayo, 1989.

[CHOV92] Chovil, Nicole. *Discourse-Oriented Facial Displays in Conversation.* University of British Columbia. Research on Language and Social Interaction. Vol.25, 1992:163-194.

[COXE94] Cox, Earl. *The Fuzzy Systems Handbook.* AP Professional Press. 1994.

[DAME98] Damer, Bruce. *Avatars!* Peachpit press. 1998.

[DAVI97] Davis Tom, Neider Jackie, Woo Mason. *Opengl Programming Guide: The Official Guide to Learning Opengl.* Addison-Wesley Pub.1997.

[EKCM75] Eckman, Paul and Friesen, Wallace. *Unmasking the Face.* Prentice Hall, New Jersey. 1997.

[FORG82] Forgy, C.L. 1982. *Rete: A fast algorithm for the many pattern/many object match problem.* Artificial Intelligence 19(1):17:37.

[FURN91] Furness, Tom. *Exploring Virtual Worlds with Tom Furness.* ACM-MEMBERNET. Julio, 1997. vol.34(7).

[GERM92] Germain, Ellen. *Introducing Natural Language Processing.* AI Expert Magazine. Agosto, 1992. vol.7(10) 30-35.

[GIAR94] Giarratano, Joseph and Riley, Gary. *Expert Systems. Principles and Programming.* PWS Publishing Company. 1994.

[HOGN98] Hogni Vilhjalmsson, Hannes and Cassell, Justine. *BodyChat: Autonomous Communicative Behaviors in Avatars.* MIT Media Laboratory. Submitted to the 2nd annual ACM International Conference on Autonomous Agents. Minneapolis, 1998.

[HUTC95] Hutchinson, Alan. *Algorithmic Learning.* Department of Computer Science, King's College London. Clarendon Press, Oxford. 1995.

[IZAR77] Izard, Carrol E. *Human Emotions.* Plenum Press, NewYork 1977.

[IZAR92] Izard, Carrol E. *Basic Emotions, Relations Among Emotions, and Emotion-Cognition Relations.* Psychological Review. Julio, 1992. vol.99(33) 561-565.

[KERS97] Kershaw, Paul. *The Ways and Means of Synchronous Computer Mediated Communication: Online Prosody and its Linguistic Relevance.* Department of Linguistics Michigan State University. Spring 1997.http://www.custom.net/ seahorse/ways.htm

[LUCA93] Luca Adriano, Maxinez David, Zapata Angel, Ferreyra Andres. *Controladores Fuzzy Logic.* Departmento de Ingeniería Electrónica. UNAM. 1993.

[MICR] http://www.research.microsoft.com.

[MULL96]Muller, Jorg P. *The Design of Intelligent Agents. A Layered Approach.* Springer. Alemania, 1996.

[NAGA94] Nagao-K. Takeuchi-A. *Speech dialogue with facial displays: multimodal human-computer conversation.* 32nd Annual Meeting of the Association for Computational Linguistics. Proceedings of the Conference. Las Cruces, NM,USA. Junio, 1994.

[NEWE72] Newell A, H. A. Simon. *Human Problem Solving.* 1972. Englewood Cliffs, NJ. Prentice Hall.

[PARK96] Parke Frederic, Waters Keith. *Computer Facial Animation.* A K Peters. Wellesley, Massachusetts. 1996.

[PIES97] Piesk J, Trogemann G. *Animated interactive fiction: Storytelling by a conversational virtual actor.* Proceedings. International Conference on Virtual Systems and MultiMedia, VSMM '97. Geneva, Switzerland. Septiembre, 1997.

[PSLA99] Perceptual Science Laboratory. Universidad de California Santa Cruz. http://mambo.ucsc.edu. Julio, 1999.

[RUSS95] Russell Stuart, Norvig Peter. *Artificial Intelligence A Modern Approach.* Prentice Hall Series in Artificial Intelligence. 1995.

[SAVA95] Savage-Carmona Jesus.*Hybrid System with Symbolic AI and Statistical Methods.* Ph.D. Thesis Dissertation. University of Washington, Seattle 1995.

[SCHA81] Schank Roger, Riesbeck Christopher. *Inside Computer Understanding.* Yale University. Lawrence Erlbaum Associates, Publishers, 1981.

[SCHM95] Schmalz-M-S. Dankel-D-D-II. *HEART-a model of emotion for natural language interpretation. 1. Background and model requirements.* Proceedings of the Eighth Florida Artificial Intelligence Research Symposium. FLAIRS-95. Melbourne, FL, USA. Abril, 1995.

[SCHM95b] Schmalz-M-S. Dankel-D-D-II.*HEART-a model of emotion for natural language interpretation. 2. Model specification and integration.* Proceedings of the Eigth Florida Artificial Intelligence Research Symposium, FLAIRS-95. Melbourne, FL, USA. Abril, 1995.

[SCHM96] Schmalz-M-S. Dankel-D-D-II. *HEART-a model of emotion for natural language interpretation. 3. Mechanism for emotional reverberation, decay, and priming.* Proceedings of the Ninth Florida Artificial Intelligence Research Symposium, FLAIRS-96. Key West, FL, USA. Mayo, 1996.

[SIMM86]Simmons Charles B. *Metaphor Interpretation in Natural Language Understanding Systems.* Master Thesis. University of Washington, Seattle 1986.

[TAKE93] Takeuchi-A. Nagao-K. *Communicative facial displays as a new conversational modality.* Human Factors in Computing Systems. INTERCHI '93. Amsterdam, Netherlands. Abril, 1993.

[WATE93]Waters K, Leyergood T. *DECFace and Automatic Lip Synchronization Algorithm for Synthetic Faces.* Technical Report CLR, 93. Digital Equipment Corporation. Cambridge.

[WHIT90]White-G-M. *Natural language understanding and speech recognition.* Communications of the ACM. vol.33, no.8. pp. 72-82. Agosto, 1990.

[WATS97]Watson, Mark. *Intelligent Java Applications.* Morgan Kauffman Publications. San Francisco, California 1997.

[WINO72]Winograd, T. *Understanding Natural Language.* Academic Press. NewYork, 1972