Inteligencia Artificial e Ingeniería del Conocimiento

- Revisión de la I.A clásica
  - ✓ Representación y búsqueda
  - ✓ **Sistemas expertos**
  - ✓ METOLOGIA KADS
- "Nuevos" enfoques de la I. A.
  - ✓ Agentes Inteligentes
  - ✓ Aprendizaje
  - ✓ Lógicas multivaluadas
  - ✓ Computación evolutiva

I. A. Clásica Sistemas Expertos

- Introducción y definición
- Etapas en el proceso de creación
- Herramientas de desarrollo
- Ejemplos
  - ✓ MYCIN
  - ✓ DENDRAL
  - ✓ PROSPECTOR
  - ✓ G2

Introducción I

- Introducción
  - ✓ Actitudes inteligentes
    - ✓ Resolución de problemas (Problemas de búsqueda)
    - ✓ Capacidad de almacenamiento de información (Problema de representación)
    - ✓ Explotación del conocimiento (Utilización de conocimiento experto)
  - ✓ Se plantea la necesidad de desarrollar sistemas optimizados de búsqueda de información
    - ✓ Bases de datos inteligentes
    - ✓ Sistemas expertos
  - ✓ Definición de sistema experto
    - ✓ Aplicación informática que es capaz de aplicar conocimiento substancial en áreas específicas de experiencia con objeto de resolver problemas.

Introducción II

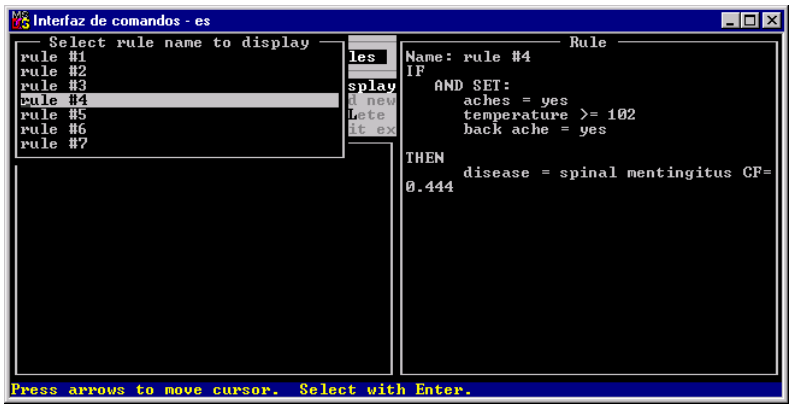
- Características de un sistema experto
  - ✓ Ser capaz de aprender de los expertos humanos
  - ✓ Mantener y actualizar los conocimientos a través de la lectura, planteamiento de cuestiones e incluso de la experiencia adquirida
  - ✓ Presentar sus conclusiones a los usuarios humanos de igual manera que un experto:
    - ✓ Justificar, clarificar y explicar su modo de razonamiento e incluso instruir al interlocutor
  - ✓ Deberá utilizar todas o algunas de esta herramientas
    - ✓ Usar reglas experimentales heurísticas para evitar la búsqueda ciega
    - ✓ Manipulado de símbolos complejos
    - ✓ Interactuar con el usuario

Ejemplos I

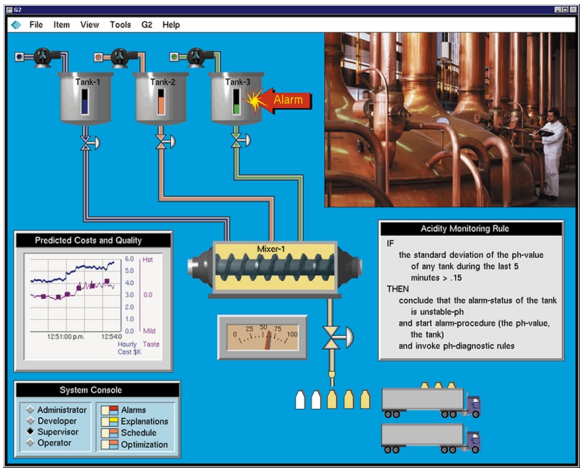
- Aplicaciones (por campos)
  - ✓ Medicina
    - ✓ MYCIN
      - Enfermedades infecciosas
    - ✓ INTERNIST
      - Diagnóstico en medicina interna
  - ✓ Geología
    - ✓ Prospector
      - Evaluación de recursos geológicos
  - ✓ Control Industrial
    - ✓ G2
      - Control en tiempo real
    - ✓ COPMA
      - Ejecución de Procedimientos de Operación de emergencia

Ejemplos II

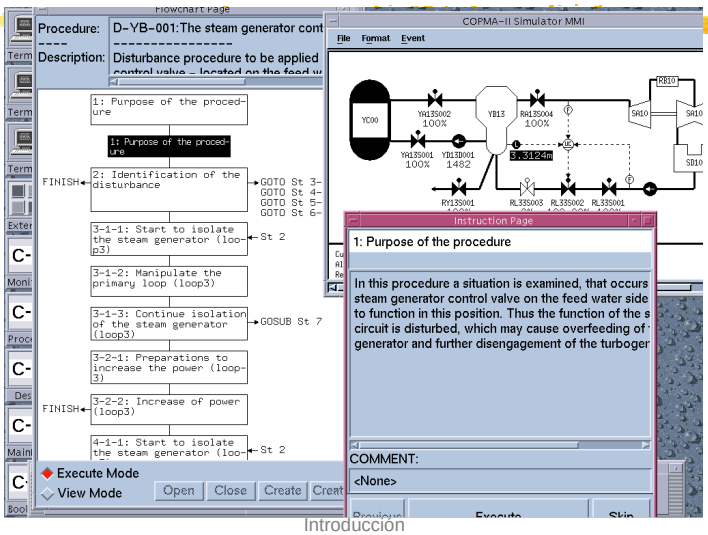
- Ejemplo 1: SUMMERS.



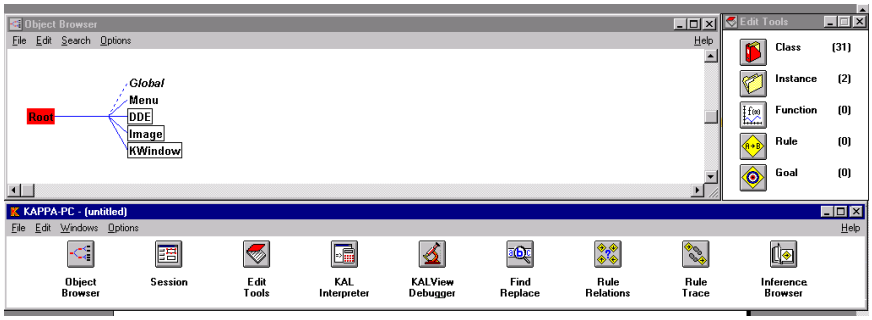
Ejemplos III: G2



Ejemplos IV: COPMA



Ejemplos (y V): Kappa



Aplicaciones I

- Control médico. Monitorizar y actuar
  - Características: Capacidad de predicción y planificación
  - Ej: VM (Medicina en UVI)
- Diseño. Dadas unas restricciones definir la configuración y características de un sistema
  - Características: Mantenimiento de todas las restricciones a lo largo del proceso de resolución. Razonamiento No-Monotónico
  - Ej: PEACE, XCON (DEC)
- Diagnóstico. Dados unos síntomas (observaciones) deducir las causas (fallos) que los provocaron
  - Características: Sistemas de razonamiento hacia atrás. Sistemas basados en modelos
  - Ej: NEAT (1989). Resolver problemas equipos de procesamiento y transmisión de datos

Aplicaciones II

- Tareas
  - ✓ Instrucción. Guiado en la educación utilizando conocimiento experto.
    - ✓ Características. Utilizan un modelo ideal de estudiante y plantean la detección y diagnóstico de los fallos que presenta
    - ✓ Ejemplos: COPMA, GUIDON
  - ✓ Interpretación. Extracción de información (simbólica) a partir de datos disponibles
    - ✓ Características. Utilizan un volumen importante de datos numéricos y han de traducirlo a información simbólica de alto nivel
    - ✓ Ejemplo: FXAA (Análisis de datos bancarios)
  - ✓ Monitorización. Detección de estados "cruciales"
    - ✓ Características. Utilizan métodos de clasificación.
    - ✓ Ejemplo: NAVEX (1984). Monitoriza transbordador espacial

Aplicaciones (y III)

- Tareas
  - ✓ Planificación. Generación de planes para conseguir un óptimo bajo unas restricciones.
    - ✓ Características: Mantenimiento de restricciones. Razonamiento no-monotónico.
    - ✓ Ejemplo: PLANPOWER (1987) Planes financieros
  - ✓ Predicción. Deducción de consecuencias a partir de situaciones.
    - ✓ Características: Razonamiento temporal. Simulación inteligente
    - ✓ Ejemplo: PLANT (1983) Simulación del efecto de plagas
  - ✓ Otros:
    - ✓ Prescripción
    - ✓ Selección
    - ✓ Simulación

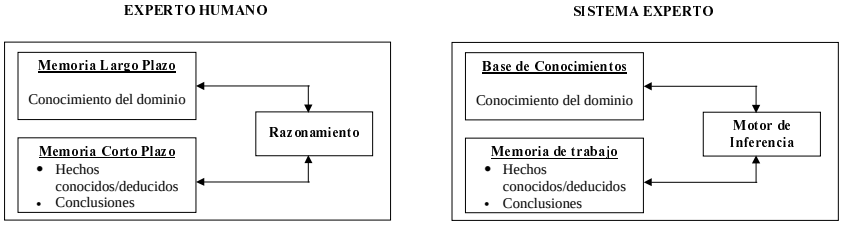
Caracterización I

- Solución algorítmica vs. Solución experta

Conventional Programs	*Expert Systems*
Numeric	Symbolic
Algorithmic	Heuristic
Information and control integrated	Knowledge separate from control
Difficult to modify	Easy to modify
Precise information	Uncertain information
Command interface	Natural dialogue with explanations
Final result given	Recommendation with explanation
Optimal solution	Acceptable solution

Caracterización (y II)

- Experto humano vs sistema experto

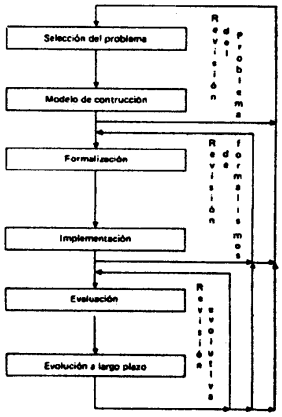


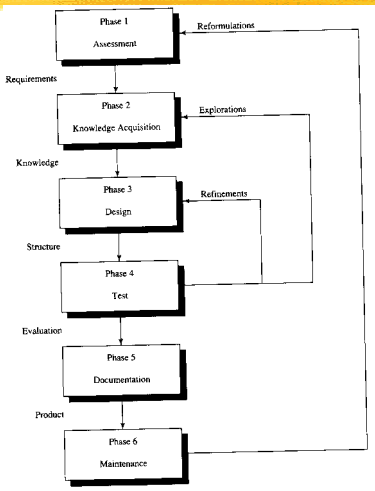
Ingeniería del Software

- Etapas en el proceso de desarrollo
  - ✓ Selección del problema objetivo
    - ✓ Investigación del problema
    - ✓ Selección del candidato
      - Interés elevado, conocimiento experto, respuestas subóptimas
    - ✓ Análisis del candidato
      - Aplicabilidad del dominio
        - Conocimiento simbólico, falta de solución algorítmica
      - Disponibilidad del experto
      - Alcance del problema
      - Análisis de costos/beneficios
        - Efectividad 70-80%
    - ✓ Selección final del candidato

Ciclo de vida. Etapas

- Ciclo de vida





Herramientas I

- Lenguajes de programación
  - Específicos de la I.A. (Prolog, Lisp)
    - Capacidad de procesamiento simbólico
    - Gestión de memoria
    - Arquitecturas específicas
    - Menor tiempo de desarrollo
  - Convencionales (C, C++)
    - Rapidez
    - Disponibilidad
    - Portabilidad
    - Mayor tiempo de desarrollo

Características físicas de las máquinas Lisp

Parámetro	Symbolics 3600	LMI Lambda	TI Explorer	Xerox 1100
Tamaño de palabra	36 bits	40 bits	32 bits	16 bits
Memoria principal	M-30MB	4-32 MB	2-16 MB	2-18 MB
Tiempo de acceso Cache	*	100 ns	*	70 ns
Tiempo de acceso memoria	200 ns	300 ns	300 ns	480 ns
Espacio direccionable	1 GB	128 MB	128 MB	32 MB
Máximo almacenamiento en disco	3.5 GB	5,15 MB	280 MB	315 MB
Comunicaciones	Ethernet	Ethernet	Eternet	Ethernet
Software básico principal	Zetalisp Common Lisp Interlisp Prolog	Zetalisp Common Lisp Interlisp Prolog Unix	Zetalisp Common Lisp Interlisp Prolog	Common Lisp Interlisp Small talk Prolog

Herramientas II

- Herramientas de desarrollo. Shells
  - Características
    - Incluyen facilidades de edición, representación, inferencia
    - Posibilidad de apoyar el diseño del interfaz
    - Existen después entornos pensados específicamente para automatizar (siempre parcialmente) la adquisición del conocimiento (OPAL, SALT, ...)
  - Origen histórico
    - Desarrollo de sistemas expertos cuyo motor de inferencia se iba ampliando a medida que lo requerían
    - Posteriormente se incorporaron módulos para hacer más fácil la escritura de las reglas, para explicar el razonamiento, para controlar y depurar la ejecución
    - Finalmente se "vaciaba" la base de conocimiento, con lo que quedaba un entorno de desarrollo

Herramientas III

- Herramientas de desarrollo. Shells
  - Ejemplos
    - EMYCIN (MYCIN)
    - OPS5 (XCON)
    - CLIPS
    - KAPPA
    - Nexpert
    - G2
    - KEE
    - ART

Shell	Sistema básico	Lenguaje	Referencia	Modelo
Age	Hearsay II	Lisp	Nii, 1979	Reglas, tablero, fuentes independientes de conocimiento
Emycin	Mycin	Lisp	Buchanan, 1984	Reglas, encadenamiento hacia atrás, consultas de diagnóstico
Expert	Casnet	Fortran	Weiss, 1984	Reglas, clasificación, consultas de diagnóstico
KAS	Prospector	Lisp	Duda, 1984	Reglas, redes semánticas hacia adelante y hacia atrás

Herramientas IV

- Shells. Características (I)
  - Potencia en la representación del conocimiento
    - Todos los entornos avanzados de hoy en día cuentan con representaciones híbridas, basadas en marcos y en reglas
    - Es deseable que la herramienta admita una red jerárquica de marcos e instancias, demonios, facetas, control de la herencia, programación orientada a objeto
    - Para la gestión de reglas debe disponer de prioridades, factores de certeza, tipos de dependencia, acceso a las agendas, etc.
  - Flexibilidad para modificar el funcionamiento original del programa y añadirle capacidades definidas por el diseñador del sistema experto.
  - Comodidad y sencillez en el manejo del programa
    - En este sentido es de gran ayuda contar con un entorno gráfico en el que haya editores y visualizadores para las reglas y los marcos, etc.

Herramientas V

- Shells. Características (II)
  - Eficiencia
    - En sistemas de cierto tamaño es especialmente importante que el sistema responda con rapidez, pues de otro modo difícilmente será aceptado por el usuario final.
  - Robustez
    - En algunas herramientas los errores de implementación, además de que hacen perder gran cantidad de tiempo al diseñador del sistema experto, impiden su utilización en entornos industriales, donde la fiabilidad es un requisito primordial.
  - Traza y depuración.
    - Ahorran tiempo y le permiten mantener la consistencia de la base de conocimiento.

Herramientas VI

- Shells. Características (III)
  - Explicación del razonamiento
    - Sería deseable que además de mostrar el encadenamiento de las reglas, diera al usuario la posibilidad de definir sus propios métodos de explicación.
  - Interfaces:
    - Con el usuario
    - Con bases de datos
    - Con lenguajes de programación, ya que en prácticamente todas las aplicaciones reales se hace necesario realizar cierta programación algorítmica, sobre todo por motivos de eficiencia.
    - Con el entorno: acceso al sistema operativo, intercambio de datos con otras aplicaciones, control del entorno de ventanas, etc.

Herramientas (VII)

- Shells. Características (IV)

	S1	KEE	Art	Loops	SRL.
Inferencia					
Encadenamiento hacia adelante	x	x	x	x	x
Encadenamiento hacia atrás	x	x	x	x	x
Herencia	x	x	x	x	x
Representación					
Reglas	x	x	x	x	x
Marcos	x	x	x	x	x
Lógica	No	x	x	No	x
Valores activos	No	x	No	x	x
Incertidumbre incorporada					
Factores de certidumbre	x	No	x	No	No
Rastreo con dependencia	No	No	x	No	No
Interfaz con el usuario					
Editor gráfico	No	x	x	x	x
Menús	x	x	x	x	x
Ventanas múltiples	x	x	x	x	x
Seguimiento	x	x	x	x	x
Explicaciones incorporadas	x	x	x	No	x

Encadenamiento hacia delante vs Encadenamiento hacia atrás

- Encadenamiento hacia delante
  - ✓ Estrategia de inferencia que a partir de un conjunto de hechos, deriva nuevos hechos utilizando reglas cuyas premisa coinciden con los hechos conocidos
  - ✓ Finalización del proceso
    - ✓ Alcanzado el objetivo
    - ✓ No existen reglas
- Relacionado con los Sist. Prod. dirigidos por datos (*Data-driven*)
- Requiere un mecanismo de resolución de conflictos
  - ✓ Selección de la regla que se ha de disparar

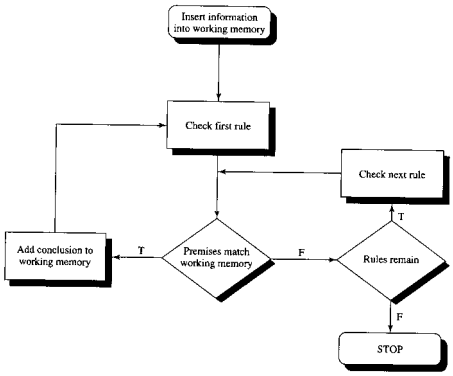
Encadenamiento hacia delante vs Encadenamiento hacia atrás

- Ejemplo de reglas

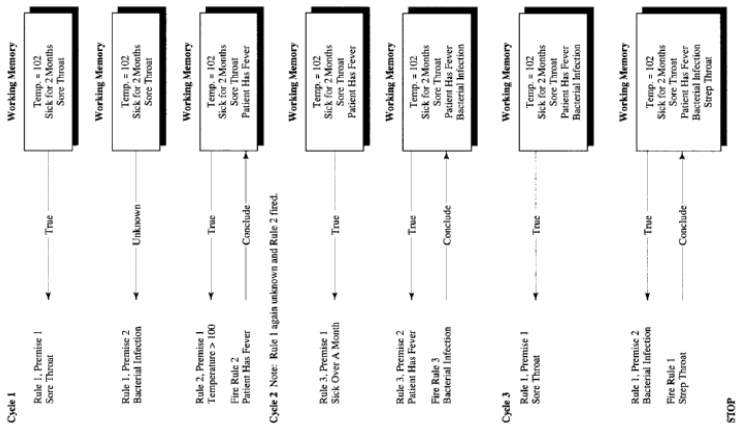
Rule 1
IF The patient has a sore throat
AND We suspect a bacterial infection
THEN We believe the patient has strep throat

Rule 2
IF The patient's temperature is > 100
THEN The patient has a fever

Rule 3
IF The patient has been sick over a month
AND The patient has a fever
THEN We suspect a bacterial infection



Encadenamiento hacia delante



Encadenamiento hacia delante

- Problemas
  - ✓ Disparo de reglas innecesarias
  - ✓ Resolución de conflictos
    - ✓ La primera regla que satisface la memoria de trabajo
    - ✓ Regla de mayor prioridad
    - ✓ Regla más específica
      - Aquella que contiene más elementos en sus premisas
    - ✓ Aquella regla que se refiere al elemento más recientemente añadido
    - ✓ Discriminar aquella regla que ya ha sido disparada
    - ✓ Disparo de reglas que origina diferentes líneas de razonamiento (Campo Viewpoint)

Encadenamiento hacia atrás

- Estrategia de inferencia que intenta probar una hipótesis a partir de sus premisas
  - ✓ Desencadena la búsqueda que tienen en sus conclusiones las premisas necesarias (Sub-goals)
  - ✓ Las premisas que no aparecen en ninguna regla: PRIMITIVAS
    - ✓ Son solicitadas al usuario

Rule 1
If There are signs of throat infection
And There is evidence that the organism is streptococcus
Then Patient has strep throat

Rule 2
If The patient's throat is red
Then There are signs of throat infection

Rule 3
If The stain of the organism is grampos
And The morphology of the organism is coccus
And The growth of the organism is chains
Then There is evidence that the organism is streptococcus

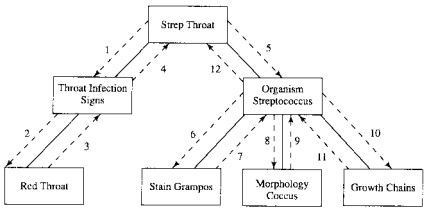


FIGURE 4.5 Backward-chaining search.

Comparación Enc. Adelante/Atrás

- EH Adelante
  - Ventajas
    - Proporcionan mucha información a partir de una pequeña cantidad de información
    - Buen enfoque (Resolución de problemas)
      - Planificación, control, monitorización, interpretación (Sistemas SCADA)
  - Inconvenientes
    - Posibilidad de hacer cuestiones inútiles, sin relación
- EH Atrás
  - Ventajas
    - Problemas que comienzan con el planteamiento de hipótesis
    - Mantienen preguntas con temas relacionados
    - Buen enfoque
      - Tareas de diagnóstico, depurado de errores
  - Inconvenientes
    - Puede tardar mucho en detectar objetivos que no se pueden conseguir

Búsqueda en profundidad vs Búsqueda en anchura

- Considerar las reglas

Rule 1
IF You purchase meat—P1
THEN You should serve red wine—C1

Rule 2
IF You purchase fish—P2
THEN You should serve white wine—C2

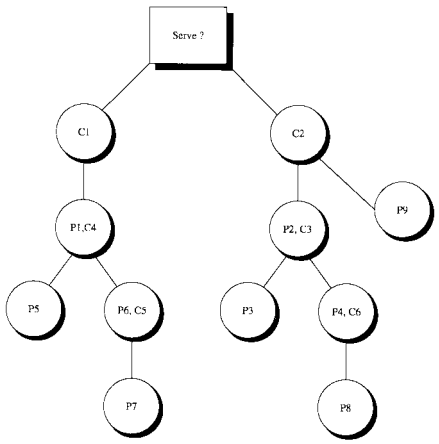
Rule 3
IF The store has fish—P3
AND You have enough money to buy fish—P4
THEN You purchase fish—C3

Rule 4
IF The store has meat—P5
AND You have enough money to buy meat—P6
THEN You purchase meat—C4

Rule 5
IF You have greater than or equal to $10—P7
THEN You have enough money to buy meat—C5

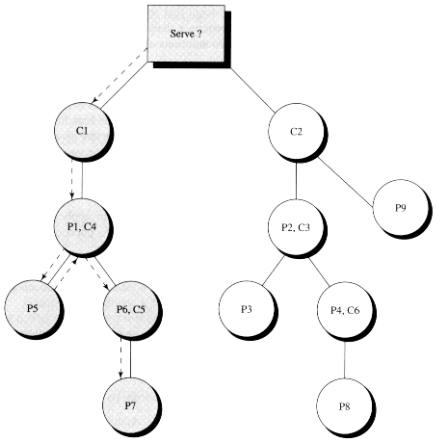
Rule 6
IF You have between $5 and $10—P8
THEN You have enough money to buy fish—C6

Rule 7
IF You really like white wine—P9
THEN You should serve white wine—C2

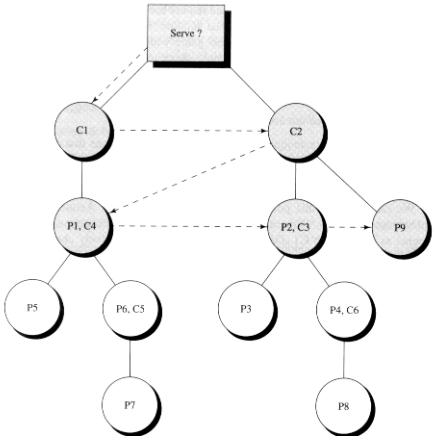


Búsqueda en profundidad vs Búsqueda en anchura

- Primero Profundidad



- Primero anchura



EMYCIN I

- Origen. MYCIN.
  - ✓ Aplicación
    - ✓ Enfermedades infecciosas (Bacterimia, Meningitis)
    - ✓ Necesidad de reacción rápida ante la presencia de enfermedades
    - ✓ Test sanguíneos: 2 dias => Necesitaba conocimiento experto
    - ✓ Tratamiento adecuado (un 66% resultaba inadecuado)
  - ✓ Desarrollo
    - ✓ 20 personas-año
    - ✓ 500 reglas
  - ✓ Características
    - ✓ Implementado en LISP (Interlisp)
    - ✓ Sistema experto basado en reglas
    - ✓ Encadenamiento hacia atrás
    - ✓ Utiliza grados de certidumbre

EMYCIN II

- Ejemplo de regla
  - ✓ Lenguaje natural

IF	El tipo de organismo es gram-negativo
AND	La morfologia del organismo es redondeado
AND	La aerobicidad del organismo es anaerobia

THEN	Existe una fuerte evidencia (0.8) de que la clase de organismo sea Enterobacteriae

  - ✓ LISP

IF	(AND (SAME CNTEXT GRAM GRAMNEG)
	(SAME CNTEXT MORPH REDOND)
	(SAME CNTEXT AIR ANAEROBIC))

THEN	(CONCLUDE CNTEXT CLASS ENTEROBACTERIAE TALLY .8)

EMYCIN III

- Principios de funcionamiento
  - ✓ Separación del conocimiento del control
    - ✓ Conocimiento
      - Conjunto de reglas sobre el dominio
    - ✓ Control
      - Encadenamiento hacia atrás
      - Búsqueda primero-profundidad
      - Incorpora metaconocimiento
        - – Manejo de reglas por otras

IF	Infección es pélvica
AND	Hay reglas que mencionan las enterobacterias
THEN	Hay evidencia de que las reglas que mencionan las enterobacterias deben ser mencionadas

EMYCIN IV

- Propiedades
  - ✓ Empleo del razonamiento inexacto
    - ✓ Factores de incertidumbre
      - -1 Falso <-> 1 Verdadero
      - Ejemplo: Evidencia puede ser verdadera (Evidencia 0.7)
  - ✓ Recuerda la sesión anterior
  - ✓ Interacción en lenguaje natural
  - ✓ "Spelling checker"

MYCIN: What is the infection
User: Endddduritis
FATAL ERROR 6B4A PROGRAM ABORTED
INVALID RESPONSE, RETYPE
MYCIN: Do you mean to type Endartitis
User: Yes

INCORRECTO

EMYCIN V

- Propiedades
  - ✓ Proporciona explicaciones: WHY, HOW

MYCIN: How old is the car's battery?
User: Why?
MYCIN: This will aid in determining the battery' condition
RULE 037
IF The car abttery's age is over 5 years
AND The car battery's voltage is below 10 volts
THEN The battery's condition is probably bad

MYCIN: I Recommend you to replace the battery
User: How?
MYCIN: I used the rule 044 to conclude that the recommendation is to replace the batery
RULE 044:
IF The battery's condition is probably bad
THEN Recommend replacing the battery
I used the rule 037 to conclude the condition of the battery
RULE 037
IF The car battery's age is over five years
AND The car battery's voltage is below 10 volts
THEN The battery's condition is probably bad

You told me

EMYCIN (y VI)

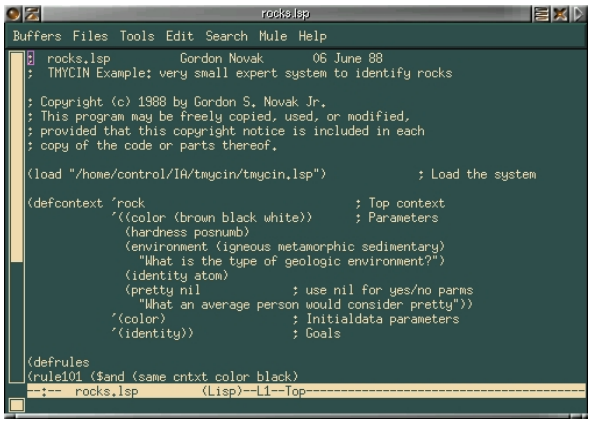
- Ejemplo. TMYCIN = Tiny EMYCIN-like Expert System Tool
- Base de conocimientos
  - ✓ Contexto
    - ✓ Estructura de datos donde se reflejan los datos acerca de un objeto (enfermedad, roca, avería). Se describen en "context definition" o Clase
    - ✓ (defcontext <context-name>
    - ✓ <parameters>
    - ✓ <initial-data>
    - ✓ <goals>)
      - <context-name> es el nombre del contexto (la clase de objeto a identificar o diagnósticar)
      - <parameters> es la lista de parámetros/características que lo describen
      - <initial-data> es la lista de parámetros iniciales que se comienzan en cada consulta
      - <goals> es la lista de valores cuyos valores son buscados como resultado de la consulta

EMYCIN. Ej: Tmycin I

- Base de conocimientos
  - ✓ Contexto. Parámetros (<parameter-name> <type> <prompt>)
    - ✓ <parameter-name> es siempre un símbolo simple (COLOR)
    - ✓ <type> puede ser
      - POSNUMB (positive number). No se chequea y se muestra al usuario su valor
      - Una lista de valores BROWN BLACK WHITE). Igual
      - NIL, induca una respuesta (YES/NO) Se examina de forma que se convierte en NO to (YES -1.0)
    - ✓ <prompt> (optional) Sirve para indicar la cuestión o comentario que se incluye al hacer la pregunta

EMYCIN. Ej: Tmycin II

- Base de conocimientos
  - ✓ Contexto



EMYCIN. Ej: Tmycin III

- Base de conocimientos
  - ✓ Reglas
    - ✓ (<rulename> <premises>
    - ✓ <conclusion>)
  - ✓ <premises> es una conjunción de condiciones agrupadas dentro de una llamada a la función $AND.
  - ✓ <conclusión> es una llamada a la función CONCLUDE (puede ser un DO-ALL para varias acciones)

```
(rule101 ($and (same cntxt color black)
               (notsame cntxt pretty yes)
               ($or (between* (val1 cntxt hardness) 3 5)
                    (same cntxt environment sedimentary)))
         (conclude cntxt identity coal tally 400)
```

EMYCIN. Ej: Tmycin IV

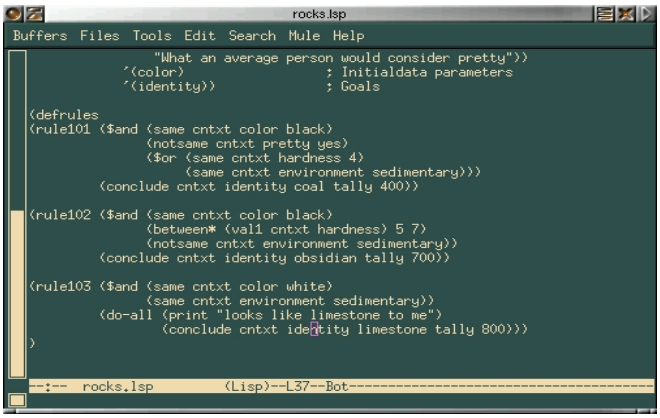
- Base de conocimientos
  - ✓ Reglas. Premisas
    - ✓ Combinaciones $AND and $OR
      - $AND. Si cualquier clausula devuelve NIL o un CF<.2 devuelve NIL. Hace un pre-análisis. En caso contrario devuelve el minimo de los CF
      - $OR devuelve el máximo de los CF de los elementos a los que afecta.
    - ✓ Operandos
      - (SAME CNTXT <parameter> <value>) chequea si un parámetro toma un valor <value> con un CF>.2 devolviendo este CF. El valor es el CF si el chequeo es (SAME CNTXT <parameter> YES).
      - (NOTSAME CNTXT <parameter> <value>) chequea si el parámetro toma el valor con un CF<=.2. Devuelve verdadero cuando el valor es "unknown"

EMYCIN. Ej: Tmycin V

- Reglas. Definición de las premisas
  - ✓ Operandos
    - ✓ (THOUGHTNOT CNTXT <parameter> <value>) chequea si el valor del parámetro tiene un CF<-.2. En ese caso devuelve el valor negado.
    - ✓ (KNOWN CNTXT <parameter>) chequea si el CF para el parámetro es >.2 (Para Yes/No CF<-.2) El valor devuelto es la unidad.
    - ✓ (NOTKNOWN CNTXT <parameter>) chequea si el CF para el parámetro NO es >.2 (Para Yes/No CF<-.2) El valor devuelto es la unidad.
  - ✓ Exploración de las reglas hacia atras con estrategia primero-profundidad. MOTOR DE INFERENCIA

EMYCIN. Ej: Tmycin VI

- Ejemplo

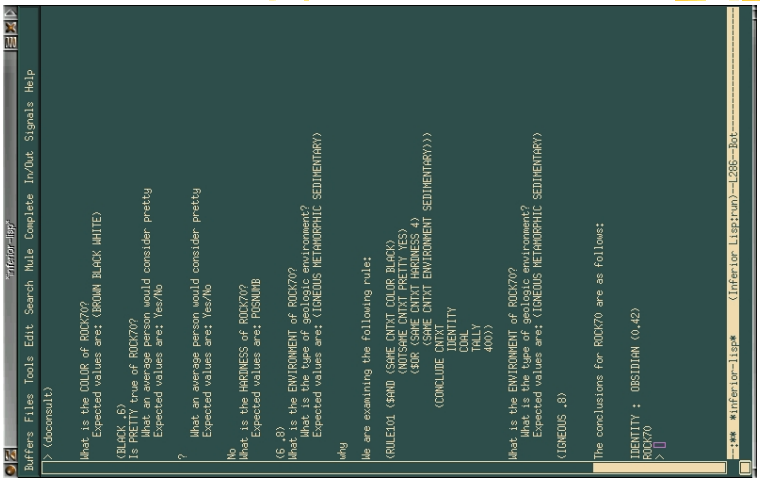


EMYCIN. Ej: Tmycin VII

- Utilización
  - ✓ Primeros pasos
    - ✓ (load "rocks.lsp")
    - ✓ (doconsult)
  - ✓ Respuestas de usuario
    - ✓ BLUE => ((BLUE 1.0)). YES/NO
    - ✓ (YES 0.6) => ((YES 0.6)).
    - ✓ ((RED 0.5)(ORANGE 0.5)). No hay análisis de consistencia
    - ✓ UNK / UNKNOWN
    - ✓ ?. El sistema coloca la "Ayuda"
    - ✓ WHY

EMYCIN. Ej: Tmycin (y VIII)

- Ejemplo de ejecución



DENDRAL

- Sistema original. Ed. Feigenbaum
- Características
  - ✓ Objetivo
    - ✓ Determinar la estructura molecular de sustancias orgánicas
    - ✓ Planificar la secuencia de reacciones para sintetizar el producto
  - ✓ Grandes espacios de búsqueda
    - ✓ Satisfacción de restricciones (precondición de reglas de producción). Impuestas por la espectrometría de masas
    - ✓ Utilización de heurísticas
  - ✓ Proyecto CONGEN (CONtraint GENerator)

If the spectrum for the molecule has two peaks at masses x_1 y x_2 such that
AND $x_1 + x_2 = M + 28$
AND $x_1 - 28$ is a high peak
AND $x_1 + 28$ is a high peak
AND at least one of x_1 or x_2 is high
CONCLUDE that the molecule contains a CETONE group

Restricciones

Heurística

PROSPECTOR I

- Sistema original. Ed. Duda. Origen de KAS
- Características
  - ✓ Objetivo
    - ✓ Evaluación de recursos geológicos para la prospección
  - ✓ Motor de inferencia
    - ✓ Sistema basado en reglas
    - ✓ Utiliza el conocimiento inexacto utilizando las probabilidades condicionadas (Teorema de Bayes)
    - ✓ Permite manejar incertidumbre con mayor grado de información
  - ✓ Dada la cláusula: E => H, se definen dos parámetros
    - ✓ Grado de suficiencia $LS = \frac{P(E/H)}{P(E/\neg H)}$
      - Representa el soporte para H dada la evidencia de E
    - ✓ Grado de necesidad $LN = \frac{P(\neg E/H)}{P(\neg E/\neg H)}$
      - Representa el descrédito de H cuando no hay evidencia de E

PROSPECTOR (y II)

- Efecto (Interpretación de los coeficientes)

LS	**Efecto en H**
'0'	H es falso cuando lo es E
Small	E es desfavorable para concluir H
'1'	E no tiene efecto para concluir H
Grande	E es favorable para concluir H
Inf	E es lógicamente suficiente para H

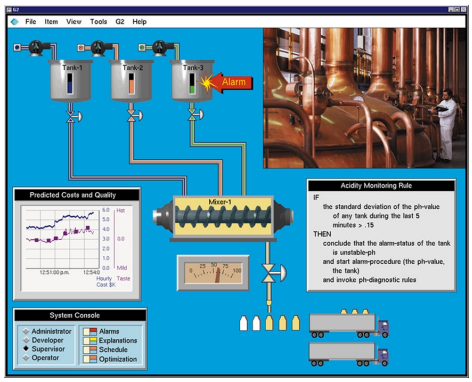
LN	**Efecto en H**
'0'	H es falso cuando E no aparece
Small	La ausencia de E es desfavorable para concluir H
'1'	La ausencia de E no tiene efecto sobre H
Grande	La ausencia de E es favorable para concluir H
Inf	La ausencia de E es lógicamente suficiente para H

Sistemas comerciales: G2 I

- Herramienta de desarrollo de sistemas expertos
  - ✓ Diferentes paradigmas de programación, Motor de inferencias,...
  - ✓ Conexión con el exterior (hardware y software)
- Utilizado en el entorno industrial
  - ✓ G2 applications can dramatically improve the performance of an operation by:
    - ✓ Continuously monitoring for potential problems before they adversely affect the operation
    - ✓ Turning complex operations data into useful information byreasoning about the data using knowledge-based models and analysis
    - ✓ Diagnosing the root cause of time-critical problems and taking the correct actions
    - ✓ Maintaining optimal operating conditions
    - ✓ Coordinating activities and information in complex operation processes

(Fuente Gensym. Co)

Sistemas comerciales: G2 II

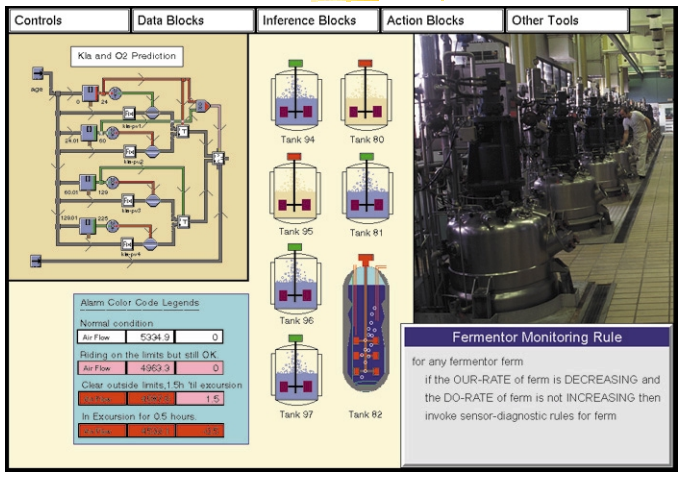


- Características de la aplicación:
  - ✓ Ejecución multithread, concurrente en tiempo real
    - ✓ Necesidad de planificación
  - ✓ Arquitectura Cliente/Servidor
  - ✓ Basado en el paradigma de POO
  - ✓ Alta capacidad de interfase
    - ✓ (Aplicaciones C/Matlab/ACSL)

Sistemas comerciales: G2 III

- Utilidades de I.A
  - ✓ Sistemas de representación
    - ✓ **Reglas**
      - Invocables hacia delante, atrás (motor de inferencia)
      - Tienen prioridad, modos de ejecución cíclicas (con tiempo de disparo)
    - ✓ Otros (marcos)
  - ✓ Manejo de incertidumbre
    - ✓ Incorporan factores de certeza
    - ✓ Variables **fuzzy**
  - ✓ Obtención de información (Percepción)
    - ✓ **Redes neuronales** para la predicción de valores, clasificación de estados

Sistemas comerciales: G2 IV



Sistemas comerciales: G2 (y V)