

Se tiene algoritmo de Dijkstra al final. Mientras no se encuentre esa lista se continua el análisis hacia el nodo final que nos interesa.

Algoritmo A*: a estrella. $f(n) = g(n) + h(n)$ al punto final
El algoritmo no dice nada estimación del costo total faltante
al respecto si hay varios \hookrightarrow peso total acumulado hasta ahora estados finales.

No se usa en contexto de varios estados finales, se usa mucho en Robótica o en donde se calculen rutas.

Busqueda con escabida de colina: se puede estancar. Decisiones locales. No necesariamente va a encontrar la óptima final.

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    current ← MAKE-NODE(problem.INITIAL-STATE)
    loop do
        neighbor ← a highest-valued successor of current
        if neighbor.VALUE ≤ current.VALUE then return current.STATE
        current ← neighbor
```

Podemos llegar a una solución que sea la óptima mejor hasta ese punto.

Elige agregar a la secuencia de pasos el mejor siguiente.

Debido a que puede atorarse. se han sugerido variantes.

Busqueda con escabida estocastica: en lugar de subir bajar e ir bajando más.

Se puede dar un empate para no atorarse en donde no va a encontrar solución.

Busqueda con escabida reiniciada aleatoriamente: no se presta para cualquier tipo de problemas. Se puede empezar en lugar - aleatorio y agregar operadores.

Algoritmo recocido simulado: aumenta y luego disminuye el costo de las operaciones. Se usa analogia con los metales. Seudo código no lo veremos en el curso.

Varia la forma en la que va explorando cada nro. del símbol es la diferencia de cada método.

Propagación de Restricciones:

Problemas de criptocanitmetica:

$$D+E=Y$$

$$N+R=E$$

$$E+O=N$$

$$S+M=M0$$

$$\begin{array}{r} \text{G G G} \\ + \text{SEND} \\ \hline \text{MORE} \\ \hline \text{MONEY} \end{array}$$

$O \leq C_1 \leq 1$
 $O \leq C_2 \leq 1$

$$\begin{array}{l} O=9 \\ S=c? \\ E=c? \\ \hline \end{array} \quad \begin{array}{l} \text{Send more...} \\ 01234567 \\ O \leq S \leq 9 \\ O \leq M \leq 9 \\ \vdots \\ S \neq M \neq R \neq \dots \end{array}$$

$T=1=C_4 \rightarrow$ al ser acero no pasa de 1 y no puede ser 0.

Propagación de restricciones: poco a poco se va acotando más los valores.
Hacer nuevas inferencias para acotar más.

$$S+M+C_3=O$$

$$S+1+C_3=O+10$$

$$S+C_3=O+9$$

$$S+C_3-O=9 \rightarrow S \geq 8$$

CROSS
ROADS
DANGER

Propagación de restricciones

Satisfacción de restricciones.

(C, S, P): Constraints, satisfaction, problem.

Problema Sub-restringido: Pone condiciones iniciales. Tiene muchas soluciones.

Problema sobre-restringido: Pone condiciones iniciales que no logran llegar a ninguna solución

Ambientes Competitivos:

Juegos de ajedrez. Juegas por turnos. En nuestro turno maximiza y en el turno del otro minimiza.

0 (max)

1 (min)

2 (max)

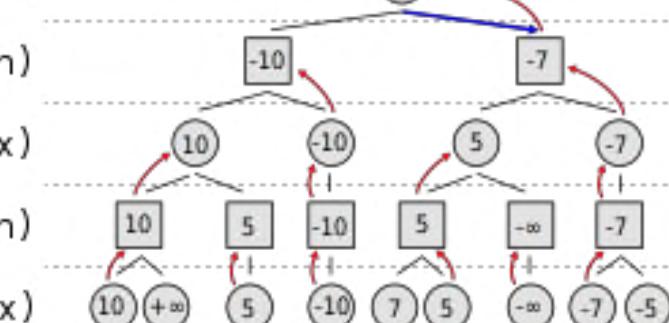
3 (min)

4 (max)

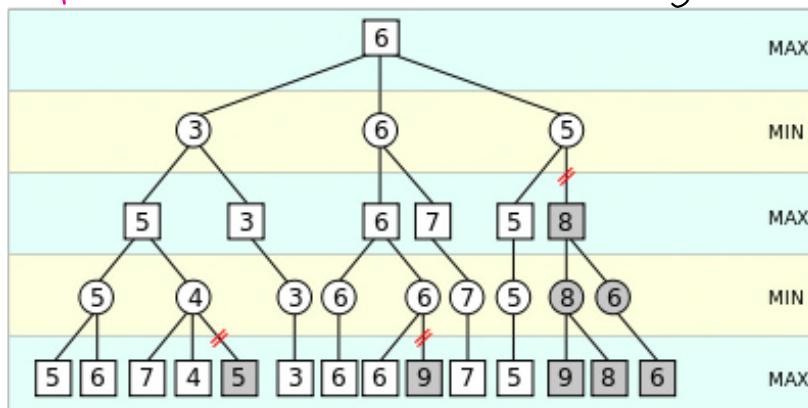
-7

-10

-7



Alpha Beta: Se descartan corte y ramas. Asignamos valores α y β en cada rama.



① $\alpha = -\infty, \beta = \infty$

② $\alpha = -\infty, \beta = \infty$

③ $\alpha = -\infty, \beta = \infty$

④ $\alpha = -\infty, \beta = \infty$

⑤ 5

⑥ $\beta = 5, \alpha = -\infty$

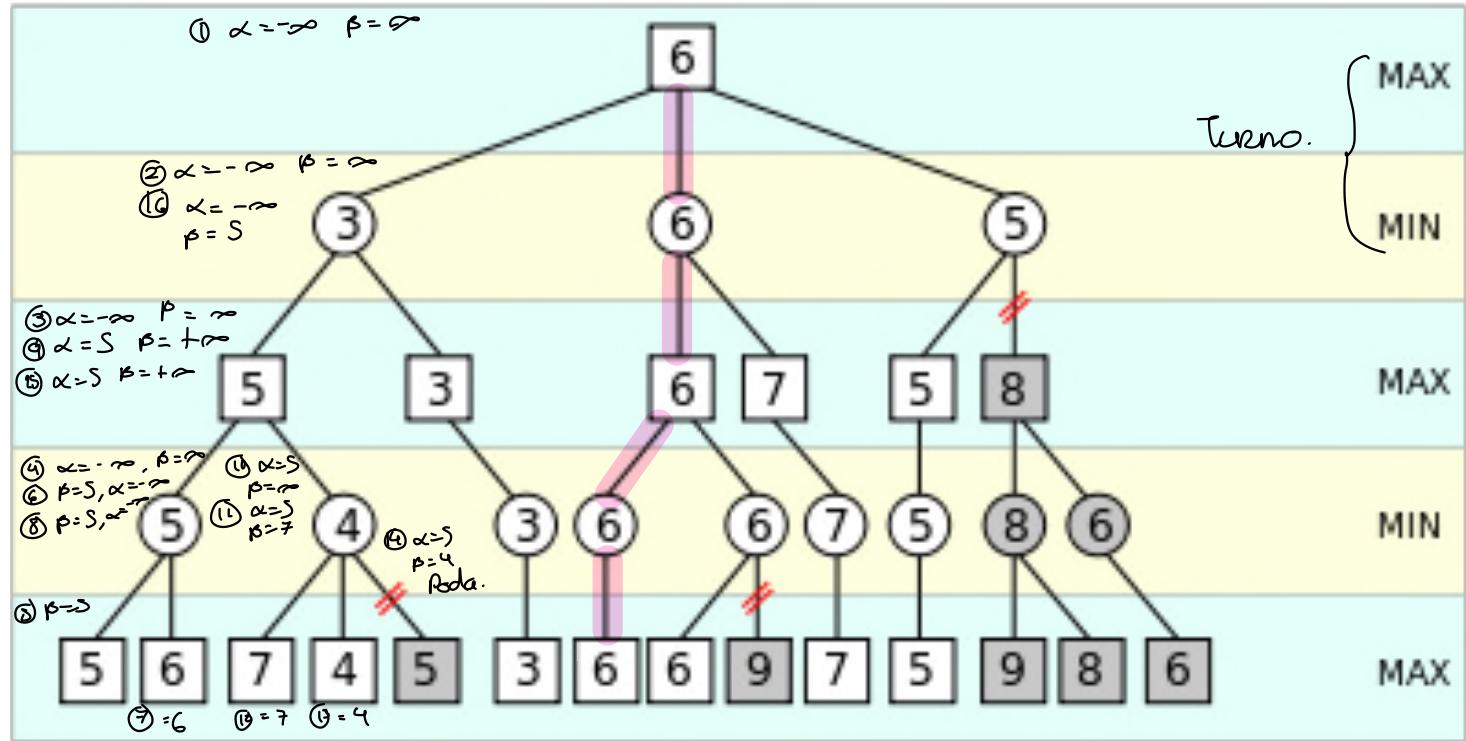
⑦ Checa el 6, pero β sigue igual

⑧ $\beta = 5, \alpha = -\infty$

⑨ $\alpha = \max$ entre $-\infty$ y lo que reciba: $\alpha = 5, \beta = \infty$

⑩ $\alpha = 5, \beta = \infty$, baja nivel hoja

Se corta cuando $\beta \leq \alpha$.



PLY: Turno que toma cada jugadore. (Copa)

25 / febrero / 19

Proyecto 2: Código que pueda jugar domino. Se tiene que poder jugar en equipo e individual.

Minimax y alfa-beta. Necesitamos investigar que tipo de información heurística necesita el domino para jugar.

Presentación oral y reporte escrito donde expliquemos nuestro código y como jugar domino.

Capacidad de meterle decisiones externas al programa. Recibir y proporcionar decisiones.

27 / febrero / 19

Marcos y maneras:

Se conocen como frames o slots.

Marcos encierran o agrupan cosas con características comunes.

Ejemplos: Marco que agrupan descripciones de coches.

Class: Transportation

Name: Audi

Origin: Germany

Esto nos recuerda a las clases y atributos de Java. Los métodos son los "procedural attachment" del marco del ejemplo.

Marcos no se sabe si son muy genéricos o muy específicos. En Java si se distinguen las instancias de las clases

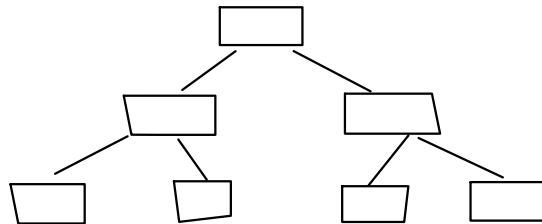
Maneras: son lo semejante a atributos en Java, son las características que

Tiene que tener los elementos en el marco.

if needed: si no está Range (el valor anterior) le pregunta al docente.

Range: Rango 1970-1995.

Jerarquía:

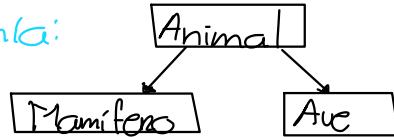


Cualquier cosa que se puede representar como un árbol

Árbol degenerado:

```
graph LR; N1[ ] --> N2[ ]; N2 --> N3[ ]; N3 --> N4[ ]
```

Taxonomía:



Se usa para tener una jerarquía clasificatoria.
La forma en la que se interpreta la forma jerárquica es lo que varía. **Is-a**.

Partonomía:

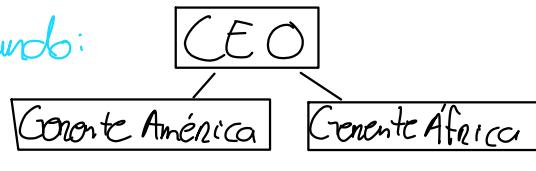


Sirven para subdividir algún elemento en sus partes.
Pant-off → leyendo hacia abajo.
Subdividir → leyendo hacia arriba.

Árbol de decisión: Ya no tenemos un concepto. Cada nodo está asociado con una pregunta. Dependiendo la respuesta elige el camino. Nodos internos = preguntas. Nodo hoja: La respuesta.

Idea de jerarquías se puede usar en diferentes dominios.

Jerarquía de mando:



Funciona para empresas.
Jefe de: si se lee hacia abajo.
Trabaja para: si se lee hacia arriba.

Árbol de sintaxis: Gramatical. Indica qué tipos de enunciados o frases es la palabra o palabras que se van juntando.

Grupos A: 3, 7, 4, 5 → Chicas superpodencosas y Urtoar
Grupo B: 1, 2, 6.

09 / marzo / 19

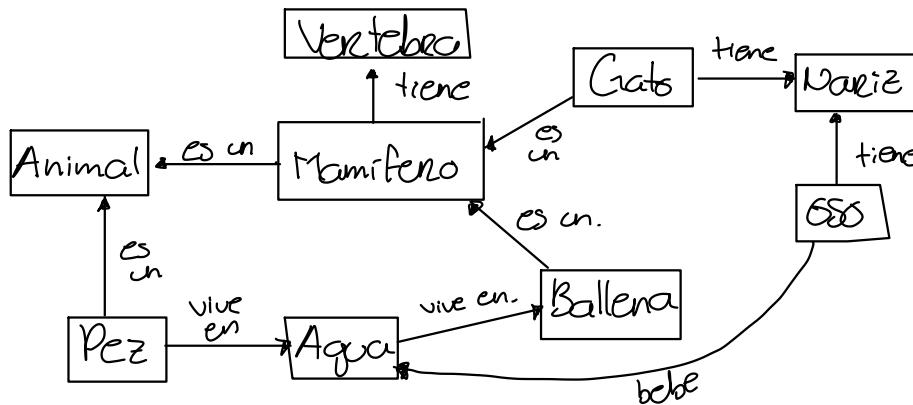
Jerarquía: conocimiento. Una de las muchas ramas la etiquetan **is a**
Taxonomía: **isa**, es un ejemplo de marcas.

Aprovechan las características de todos.

Parte jerárquica, pero no es para porque la semántica no es la misma
Relación que no es estática.

Marcos pueden ser abstractos o detallados.

Redes Semánticas.



Hay que etiquetar las relaciones para que se entiendan lo más claro posible. Conexiones deben tener sentido para llegar de un lugar a otro.

Semántica: se permiten múltiples tipos de conexión.

A lo que más se parece es a un **grafo** en estructuras de datos.

Proyecto C4C: Doug Lenat. Se tardaron 10 años y no lograron nada hasta otros 10 años. Después de 20 años se sigue construyendo la red semántica. Conectar conceptos entre sí es difícil.

Dependencia Conceptual (CD):

Capturar la esencia de las acciones que pueden aparecer en el lenguaje natural. De la vida normal. Cosas muy generales.

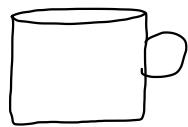
Si dos frases distintas describen la misma acción, en CD, se deben interpretar de la misma manera.

Hacer explícitas acciones implícitas

Se puede adaptar cualquier acción primitiva (cayeron) para usar CD.

Forma genérica de captar información. Describen objeto, estado. No importa si es en español o inglés porque debe representarse una sola vez.

Vaso / Glass / Cup



Típos de análisis de los verbos que describen día a día.

Atrans: transferencia abstracta. Para representar cualquier verbo que implica cambio de posesión de algo o control.

PTrans: verbos ir, poner. Implica trasladarse. Cambio físico. Movimientos reales y no abstractos.

Propel: Cualquier cosa que implique darle movimiento a alguna cosa que no se mueve por si sola. Lanzar, patinar, frenar.

Move: Solo mover.

grasp: agarrar, tomar, dejar ir, lanzar.

Ingest: ingerir. Tomar o beber. Comer, fumar, respirar. Líquidos, gas y comida.

Expel: expulsar del cuerpo. Sudar, escupir, llorar.

- mtrans:** mental transferencia. Olvidar, aprender. Tomar decisiones, informar. conscious processor → short term memory. No dura para siempre. Long term memory: se queda más tiempo. Transferencias mentales de información.
- in build:** Construir algo mental. Decidir, concluir, imaginar, considerar.
- speak:** Decir, tocar música, emitir sonidos. Ronronear, gritar.
- attend:** Poner atención con alguno de los sentidos a algo. Tocar, sentir, escuchar, mirar.

Estados.

Salud: muerto, tolerable

anticipación: nervioso, atemorizado

mental: roto, deprimido

físico: dañado, bien, perfecto

conciencia: muerto, inconsciente, dormido, despierto.

Ejemplo: visita a un restaurante:

- Attend hacia restaurante
- Ptrans hacia restaurante.
- SPEAK a maître. → mtrans de cliente a maître de deseos / # clientes.
- Ptrans hacia mesa.
- Propel silla (p' sentarse) → optativa
- Move hacia posición sentada
- Ptrans menú hacia cliente o mesa.
- Grasp menú.
- Attend detalles del menú.
- Mbuild orden.
- Speak orden hacia mesero.
- C - Attend teléfono.
- Ptrans comida de cocina hacia cliente.
- Grasp cobertos.
- Ingest comida.
- Speak petición de cuenta hacia mesero.
- Ptrans cuenta hacia cliente.
- Attend hacia cuenta.
- Mbuild decisión propina y pago.
- move mano hacia medio de pago.
- atrans dinero hacia restaurante.
- Ptrans hacia la puerta.

Algunas acciones son optativas. No aplica para todos los restaurantes. Hay variantes.

Dominio: Sobre 2 extremos. Empieza la mitad del 6 y si no la pieza más grande.

Secuencia de acciones: visitas o acciones típicas que se repiten en la vida cotidiana.

Gujones:

A parte de la secuencia de pasos se suelen poner los roles que se toman. (Cliente, mesero, cajero, etc.) También se escriben los elementos necesarios en la acción (dinero, comida, menu, etc.) Específica para qué es necesariamente. Especifica los resultados de seguir los pasos.

Se divide en escenas: Entrada. Se tiene que especificar el actor, la acción, el objeto y la dirección.

Casos:

Conocimiento que captura ejemplos o experiencias específicas. Los casos pueden ser quienes específicos como el caso de visita biblioteca o visita restaurante.

Cada caso debe contener por lo menos 2 tipos de información.

Caso:

- | | |
|--|---|
| ① Descripción del problema | Si tenemos muchos casos, el conocimiento que nos va a ayudar a filtrar el caso que queremos son las descripciones del problema y la solución. |
| | ② Descripción de la solución |
| ③ Contexto, alcance
④ Resultado (funciona o no)
⑤ Observaciones/expli-
caciones sol.
⑥ Pasos de razón que
llevan a la sol.
⑦ Observaciones/expli-
caciones sobre pasos. | ⑤ Observaciones/expli-
caciones sol. |
| | ④ Resultado (funciona o no) |
| | ③ Contexto, alcance |
| | ⑥ Pasos de razón que llevan a la sol. |
| | ⑦ Observaciones/expli-
caciones sobre pasos. |
| | ⑤ Observaciones/expli-
caciones sol. |
| | ④ Resultado (funciona o no) |

Memoria de Casos: o base de casos, se tienen varios. ¿Cuántos? → Los que sean necesarios para darle cobertura al dominio.

Kritik: diseña dispositivos electro mecánico

¿Cómo organizan la memoria de datos? Si son 3 o 4, organización plana. Si son millones si es mejor ordenar con índices o árbol de decisión. O jerárquica, etc.

Toda la información relevante.

Se especifica un nuevo problema y se hace un match para ver otros problemas

Diseño de estructura para edificios altos: decisiones sobre ubicación, lozas, tipos de diseño. En general es a 3 niveles jerárquicos.

FBS: Function Behaviour Structure

Casos siempre están representados el problema y la solución. La solución es el conjunto completo de atributos y valores y el problema es un subconjunto. La idea es reusar elementos de los Casos para futuros problemas

11 / marzo / 19

Reglas.

Sirven para representar conocimiento procedual o causal.

Ejemplos: sistema inteligente que maneja un coche de transmisión manual.

Se representa mediante tabla de precondiciones y post-condiciones.

Otra forma es con flechas: precondiciones → post-condiciones.

Otra manera es serie de reglas con condiciones tipo IF y AND.

Para metarazonamiento es mejor tener en claro las condiciones necesarias para que se cumplan las reglas.

Tipos de inferencia:

Según Umberto Eco:

- ① Deducción
- ② Inducción
- ③ Abducción

Deducción: Si sabemos $A \rightarrow B$ y sabemos A, entonces concluimos (inferimos) que B es cierto.

Concluir a partir de hechos y reglas que algunos otros hechos son ciertos (y estos deben aparecer del lado derecho de la(s) regla(s)).

"Encadenamiento hacia adelante"

Inducción: Si sabemos $A \rightarrow B$ y sabemos B, entonces concluimos (inferimos) que A puede ser cierto. Ya que también podría ser cierto que $C \rightarrow B$.

Concluir a partir de hechos y reglas que algunas otras reglas son probablemente ciertas (y estos deben aparecer del lado izquierdo de la(s) hecho(s)).

Abducción: Proponer nuevas reglas a partir de hechos / observaciones. No hay tanta certeza.

Tipos de razonamiento (tipos de problema):

Ejemplos:- Planeación }
- Diagnóstico } En General difieren en el marco entrada - salida.
- Diseño }

Son genéricos

Métodos de razonamiento

- Razonamiento basado en casos.
- Razonamiento basado en reglas.

Método de razonamiento = estrategia.

Difieren en: a) El tipo de conocimiento que requieren y b) Los pasos de razonamiento que se llevan a cabo.

13 / marzo / 19

Razonamiento basado en reglas:

$$\begin{array}{l} A \rightarrow B \\ B \rightarrow C, D \\ A \rightarrow E, F \\ C, F \rightarrow G \\ H \rightarrow I \\ C \rightarrow B \end{array}$$

Pueden ser muchas post-condiciones o precondiciones.

Prolog: conclusión (X): — \leftarrow — A
— \leftarrow — B
— \leftarrow — C

Siguiendo encadenamiento hacia delante:

Si E es cierto ¿cuál(es) regla(s) son aplicables?

Ninguna, con el conocimiento que tenemos.

Si C es cierto ¿cuál(es) regla(s) son aplicables?

$C \rightarrow B$, segundo paso $B \rightarrow C, D$.

Si A es cierto ...

$A \rightarrow B$ primero o $A \rightarrow E, F$, segundo paso $B \rightarrow C, D$

Si hay más de una regla aplicable (Problema subresolución)

Hay que hacer "resolución de conflictos".

Posibles estrategias:

- Escoger la 1^a regla aplicable

Ventaja: rápido, Desventaja: No usa conocimiento del dominio. Sesgando hacia reglas que se causalidad estén más cerca.

- Escoger la que te acerca más a la solución (escalamiento)

Ventaja: Son más acertado, Desventaja: Requiere conocimiento y tiempo.

- Escoger aleatoriamente.

Ventaja: rápido, Desventaja: (no determinístico), no usa conocimiento del dominio.

- Escoger de la mayor número de posibilidad.

Ventaja: Abre más posibilidades, Desventaja: tiempo.

Si no hay reglas aplicables: (Problemas subresoluciones)

- Regresar a caminos previamente descartados

Ventaja: no siempre queda atonado, Desventaja: lento, mucha memoria.

- Relaxación de precondiciones

Ventaja: no siempre se queda atonado, Desventaja: Solución no necesariamente óptima. Cambia conocimiento

Lenguaje OPS5 / OPS 83 → Plataforma de programación que saben que se usan reglas y solo escoges el método ya que tiene cosas preprogramadas.

Arquitectura / Lenguaje de programación para crear razonamiento basado en reglas

Sistemas Expertos:

Sistemas inteligentes basados en reglas cuyo desempeño es comparable con el de un especialista humano.

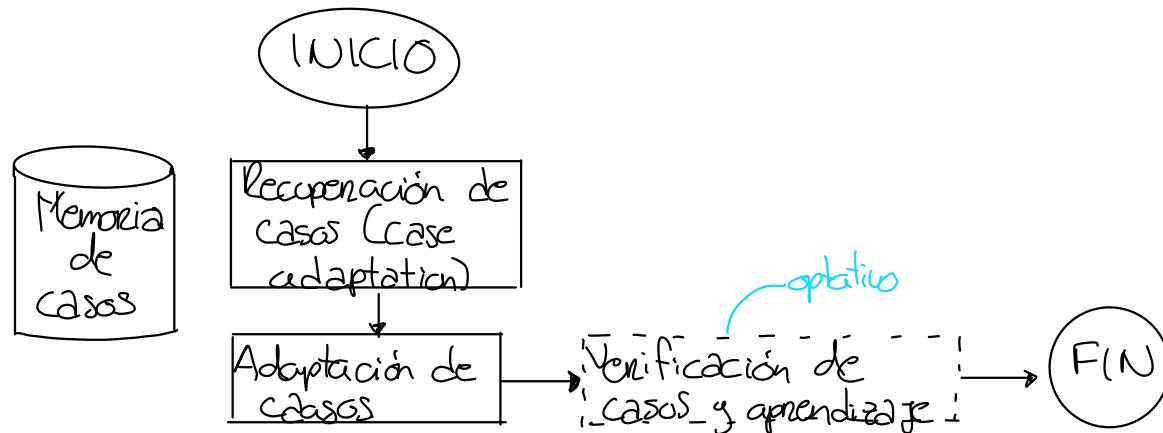
Historia:

- Finales de los años 60: Dendral, Universidad de Stanford, ED Feigenbaum
Dendral se especializa en análisis espectroscópicos de moléculas desconocidas para predecir su estructura molecular.
- Finales de los años 60 y principios de los 70: MACSYMA, MIT,
Se especializa en la solución de problemas matemáticos complejos de forma simbólica.
- Mediados de los años 70: MYCIN, U. Stanford
Se especializa en diagnosticar enfermedades bacterianas de la sangre.
- 1978: PROSPECTOR, U. de Stanford.
Se especializa en buscar minerales o petrolíferos
- Finales de los años 70: molgen, U. de Stanford.
Se especializa en diseñar experimentos para determinar la estructura de moléculas complejas
- 1981: R1 / XCON, U. Carnegie - Mellon / D.E.S
Se especializa en configurar componentes y hardware que los clientes de D.E.C. compraban.
- Mediados de los años 90: Comenzó una explosión en la cantidad de S.E y la variedad de dominios

Domino: decide con qué ficha empieza desde afuera.

20 / marzo / 19

Razonamiento basado en casos:



Razonamiento basado en casos crea diferentes casos y una solución para cada uno de ellos.

2 componentes: Problema y solución. En el ejemplo la solución puede ser darse por vencido, igualar la apuesta o subir la apuesta.

Sistemas basados en conocimiento.

Sistemas inteligentes cuyo desempeño es comparable con el de un experto humano pero cuyo conocimiento (principalmente) no está representado usando reglas.

En los dos casos hablamos de sistemas con desempeño. Sistema experto con conocimiento basado en reglas.

Tener conocimiento y usarlo.

Como primer criterio checa que coincidan origen y destino

Como segundo criterio checa que coincidan al revés.

Como tercer criterio busca que coincidan y agrega pedazos a extremos

Último módulo es un selecto de métodos. Primero busca en mapa y si no funciona busca otros.

Recursividad.

Task Structure Diagram B. Chandrasekaran.

Routar no tiene casos de inicio pero si es un sistema de razonamiento basado en casos.

03/abril/19

Razonamiento basado en casos: Recuperación de casos y adaptación de casos.

Razonamiento basado en casos y modelos.

Kritik: "el diseñador". Sistema para diseño, no es planeación. Sistema basado en casos. Cada caso es sistema diferente. Capaz de diseñar dispositivos electromecánicos. (Timbre, ___, ___)



información de elementos y como se conectan entre sí. Tienen aún más conocimiento. Estructura de dispositivo.

Kritik es el motor espacial Hubolt. Rediseño basado en modelos.

Tarea de rediseño en partes. Verificar modificaciones. Bases se basan en el modelo de Kritik. Todos los modelos siguen el mismo esquema. Modelo SBF: structure behavior function.

Comportamiento es descripción de cosas que causan otras. Mecanismo causal.