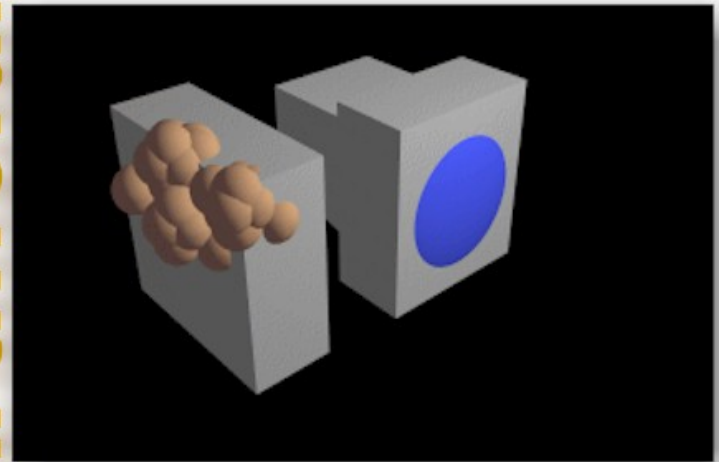


Tema 2: Estrategias de búsqueda no informada



Inteligencia
Artificial



Objetivos

- Conocer el concepto de agente inteligente y el ciclo de vida "percepción, decisión y actuación".
- Adquirir las habilidades básicas para construir sistemas capaces de resolver problemas mediante técnicas de IA.
- Entender que la resolución de problemas en IA implica definir una representación del problema y un proceso de búsqueda de la solución.
- Conocer la representación de problemas basados en estados (estado inicial, objetivo y espacio de búsqueda) para ser resueltos con técnicas computacionales.
- Conocer las técnicas más representativas de búsqueda no informada en un espacio de estados (en profundidad, en anchura y sus variantes), y saber analizar su eficiencia en tiempo y espacio.

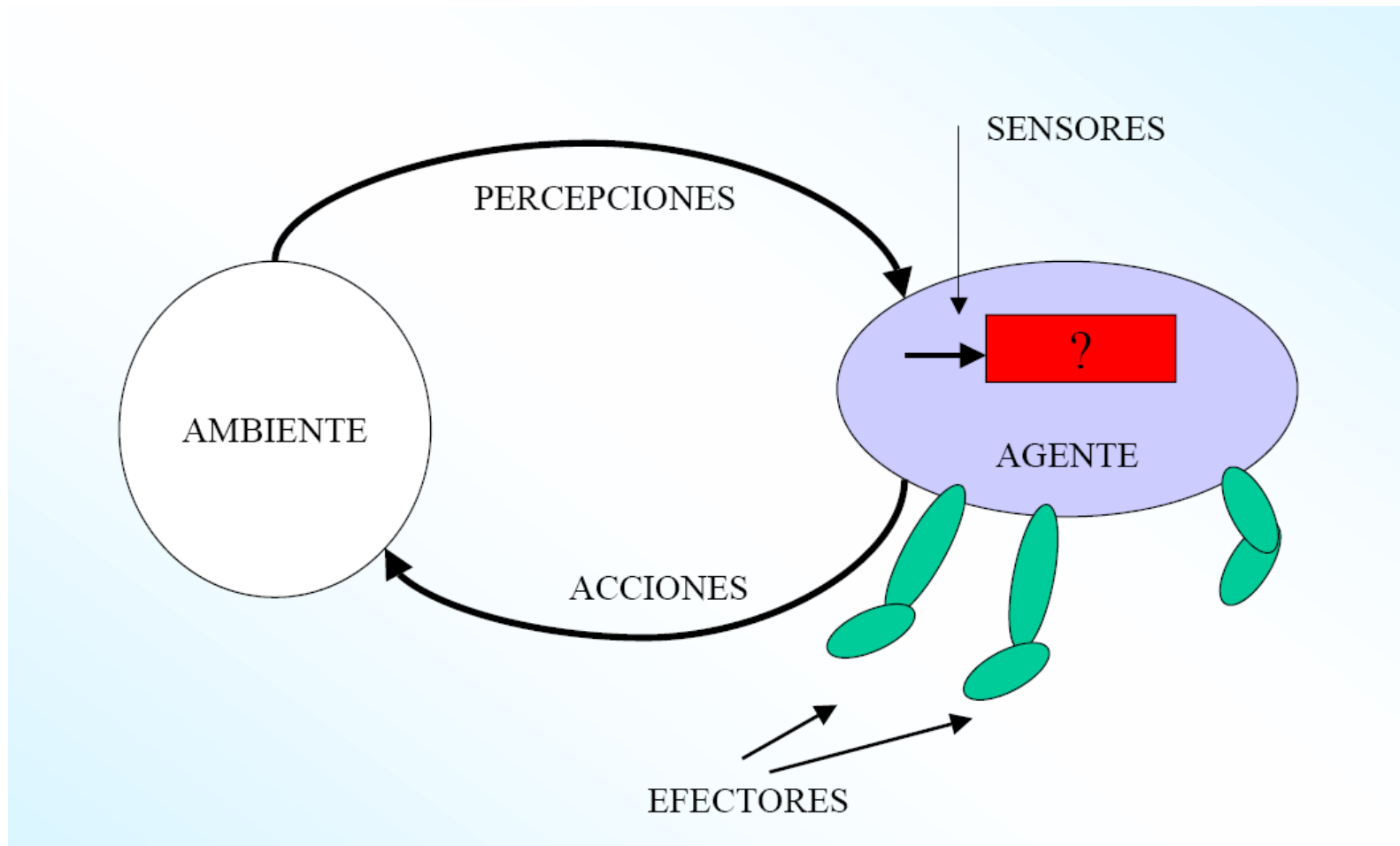
Estudia este tema en...

- Nils J. Nilsson, “*Inteligencia Artificial: Una nueva síntesis*”, Ed. Mc Graw Hill, 2000. pp. 17-32, 63-74, 103-122, 147-162

Contenido

- Diseño de un agente reactivo: arquitecturas de agentes
- Agentes reactivos con memoria
- Diseño de un agente deliberativo: búsqueda
- Ejemplos
- Búsqueda sin información
- Problemas descomponibles y búsqueda

Agentes inteligentes (racionales)



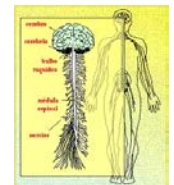
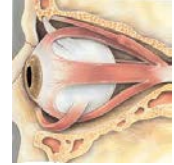
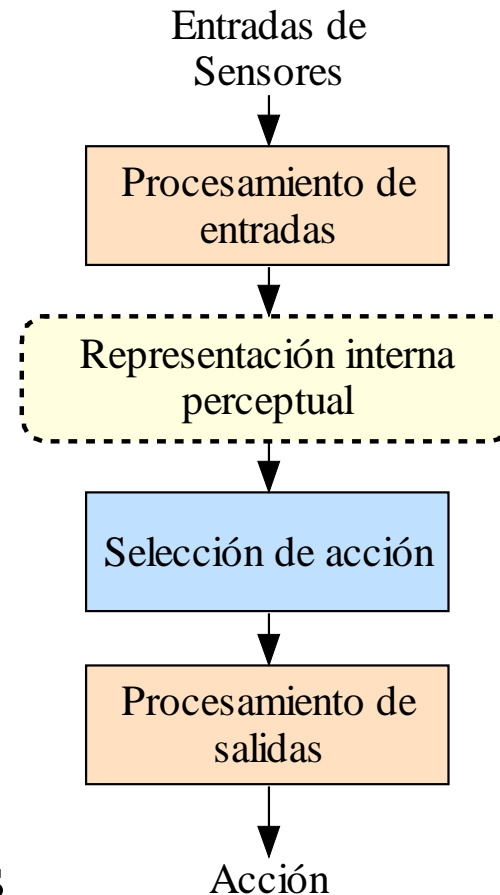
Tipos de agentes

- Agentes reactivos
 - Agentes deliberativos
 - Agentes híbridos
-
- ☐ Agentes situados en mundos habitados por otros agentes.
 - ☐ Sistemas Multiagente

Diseño de un agente reactivo

• Percepción y Acción:

- El agente reactivo percibe su entorno a través de sensores.
- ❖ Procesa la información percibida y hace una representación interna de la misma.
- Escoge una acción, entre las posibles, considerando la información percibida.
- Transforma la acción en señales para los actuadores y la realiza.



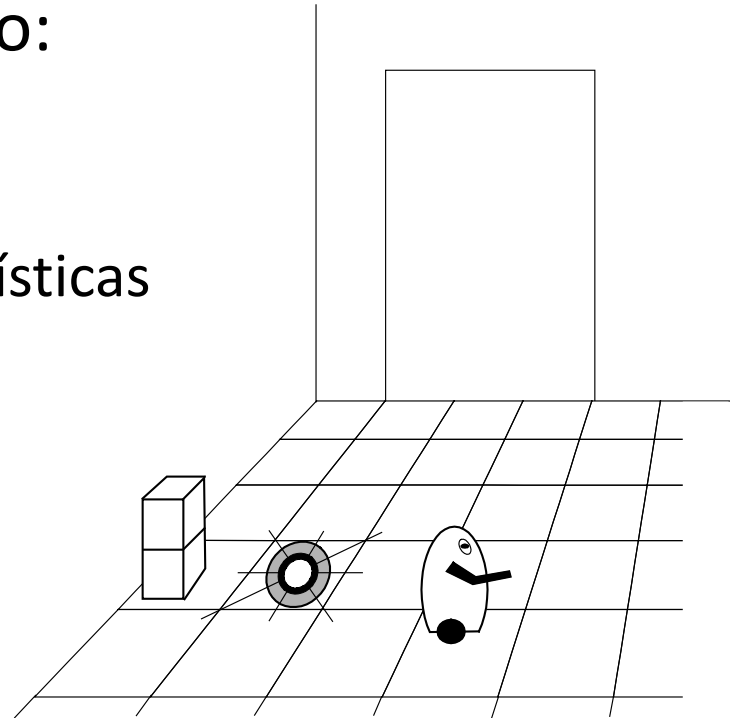
Diseño de un agente reactivo

• Ejemplo:

- Supongamos un robot en un mundo dividido en cuadrículas.
- El robot puede percibir si las 8 casillas vecinas están libres o no, con un sensor s_i por cada casilla i .
- El objetivo del robot es ir a una pared y seguir su perímetro indefinidamente.
- Tiene 4 posibles movimientos (de 1 casilla cada uno): Ir a Norte, Sur, Este u Oeste.
- No se permite que el entorno contenga pasillos estrechos (aquellas casillas rodeadas por dos o más obstáculos a ambos lados).

Representaciones del mundo

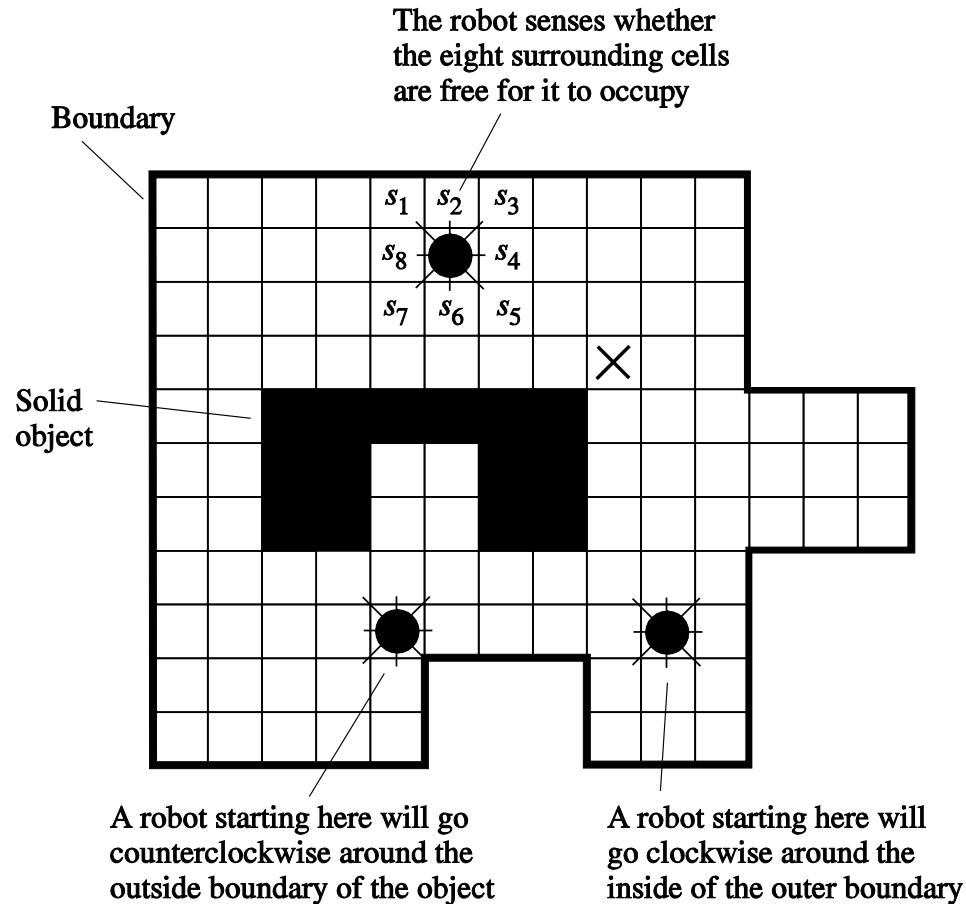
- Representaciones del mundo:
 - modelos icónicos
 - modelos basados en características



© 1998 Morgan Kaufmann Publishers

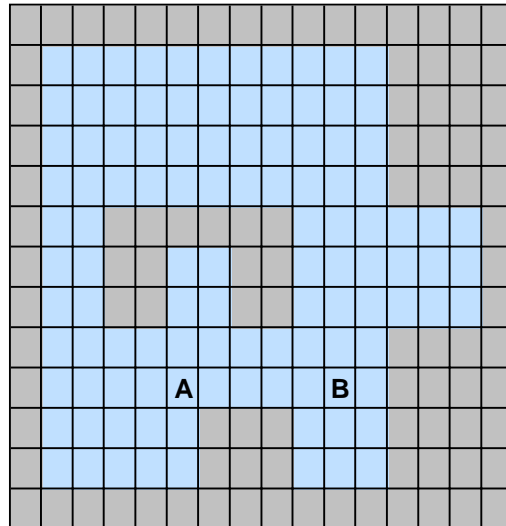
El mundo espacial cuadriculado

Diseño de un agente reactivo




© 1998 Morgan Kaufmann Publishers

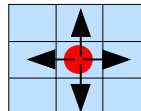
Representación



Sensores:

S_1	S_2	S_3
S_8		S_4
S_7	S_6	S_5

Movimientos:



Usaremos un vector de 8 componentes.

Cada componente i vale 0 si el sensor s_i no detecta obstáculo y vale 1 si lo detecta.

Ejemplo posición **A**:

$A =$

0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

Movimientos posibles

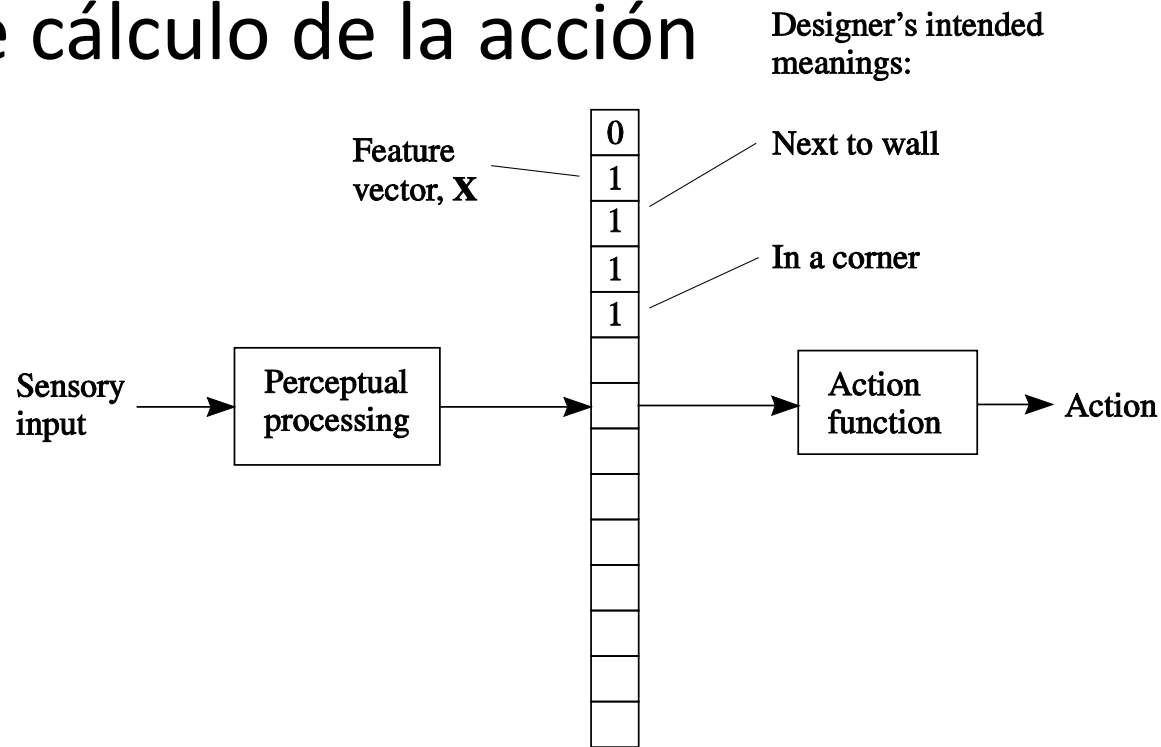
- NORTE: mueve el robot una celda hacia arriba
- ESTE: mueve el robot una celda a la derecha
- SUR: mueve el robot una celda hacia abajo
- OESTE: mueve el robot una celda a la izquierda

TRABAJO DEL DISEÑADOR:

desarrollar una función definida sobre las entradas sensoriales que seleccione la acción apropiada en cada momento para llevar a cabo con éxito la tarea del robot.

Proceso en dos fases

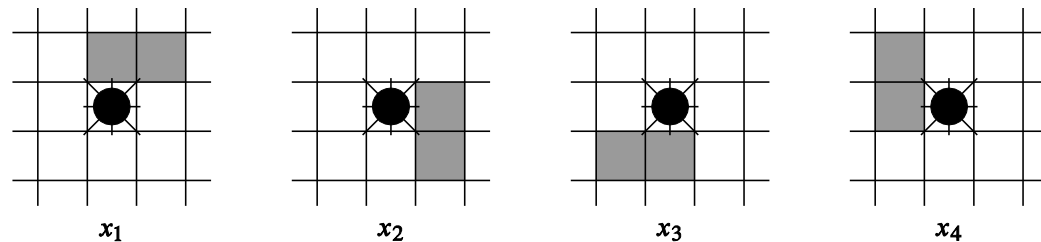
- Procesamiento perceptual
- Fase de cálculo de la acción



© 1998 Morgan Kaufmann Publishers

Percepción y acción

- Percepción:



In each diagram, the indicated feature has value 1 if and only if at least one of the shaded cells is *not* free.

© 1998 Morgan Kaufman Publishers

- Acción:

- si todas las características son cero, moverse al norte
- si $x_1=1$ y $x_2=0$, moverse al este
- si $x_2=1$ y $x_3=0$, moverse al sur
- si $x_3=1$ y $x_4=0$, moverse al oeste
- si $x_4=1$ y $x_1=0$, moverse al norte

Algunas técnicas para agentes reactivos

- Agentes en una tabla,
- Sistemas de producción,
- Redes Neuronales,
- Arquitecturas de subsunción,
- Etc.

Sistemas de Producción

$$c_1 \rightarrow a_1$$

$$c_2 \rightarrow a_2$$

...

$$c_i \rightarrow a_i$$

...

$$c_m \rightarrow a_m$$

en donde C_i es una función booleana definida sobre el vector de características, habitualmente una conjunción de literales booleanos.

Sistemas de Producción

- **Ejecución del sistema de producción:**

1. Se selecciona la primera regla y se comprueba si se cumple su condición. En caso contrario, se continúa con la siguiente hasta que se encuentre una regla con condición con valor 1.
2. La acción de la primera regla encontrada cuya condición sea 1 es la que se ejecuta. Su acción puede ser:
 - 2.1. La ejecución de una o varias acciones primitivas
 - 2.2. Una llamada a otro sistema de producción
3. **Acción por defecto:** La última regla de producción del sistema debe ser del tipo $1 \rightarrow A$, para ejecutar una acción en caso de que ninguna de las reglas anteriores cumpla su condición de ejecución

Tarea de seguimiento de bordes

Ejemplo de proceso sin fin

$x_4\overline{x_1} \rightarrow$ norte

$x_3\overline{x_4} \rightarrow$ oeste

$x_2\overline{x_3} \rightarrow$ sur

$x_1\overline{x_2} \rightarrow$ este

$1 \rightarrow$ norte

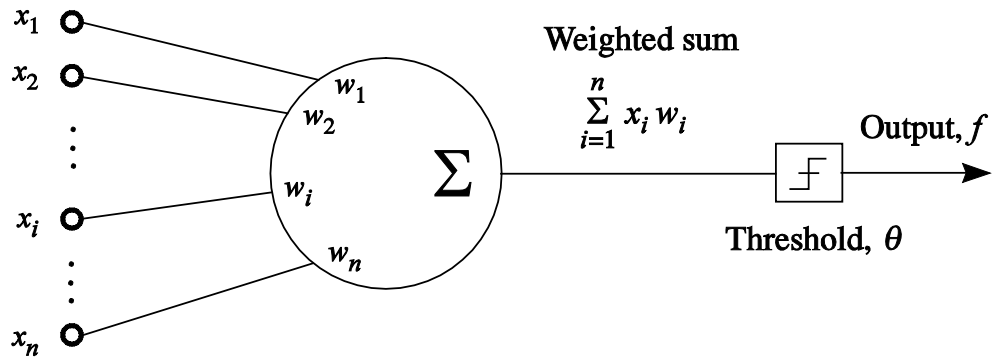
Tarea llevar al robot a una esquina cóncava

$c \rightarrow \text{nil}$

$1 \rightarrow \text{s-b}$

Ejemplo de proceso con objetivo

Redes

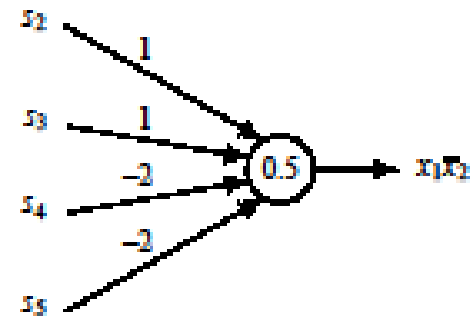


$$f = 1 \text{ if } \sum_{i=1}^n x_i w_i \geq \theta$$

$$= 0 \text{ otherwise}$$

© 1998 Morgan Kaufmann Publishers

Unidad Lógica con Umbral



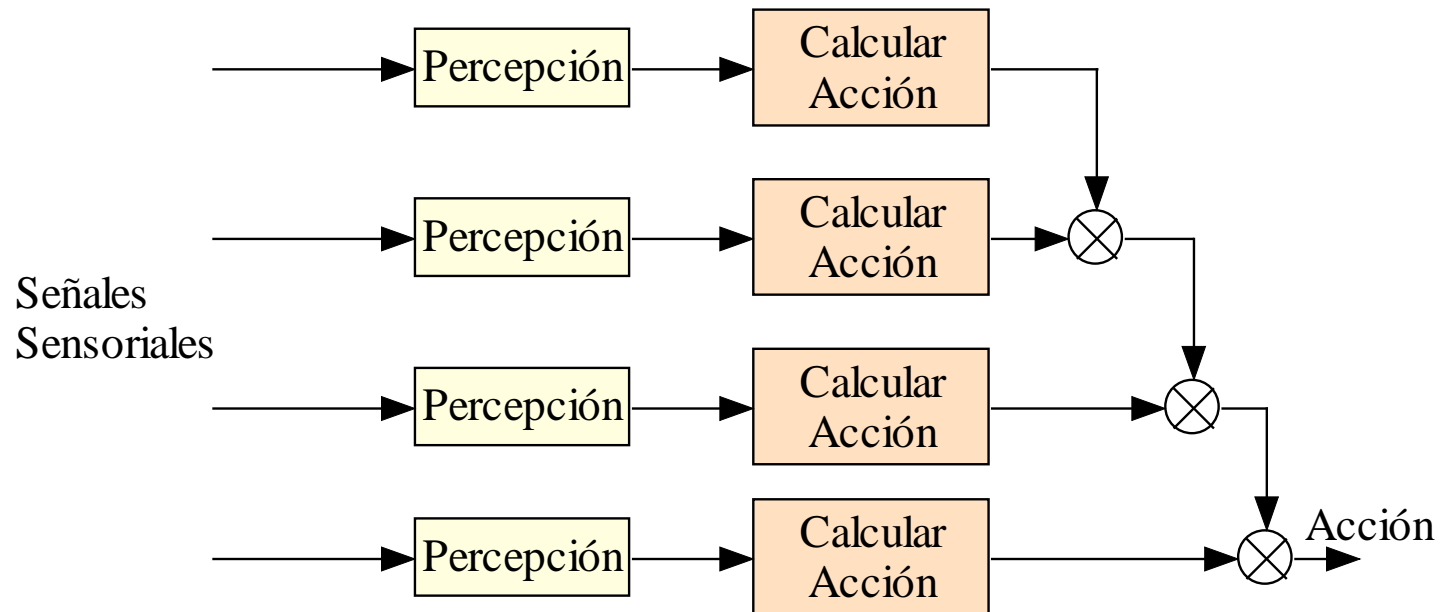
© 1998 Morgan Kaufmann Publishers

Red neuronal: red de unidades lógicas con umbral

Arquitectura de subsunción

- **La arquitectura de subsunción** consiste en agrupar **módulos de comportamiento**.
- Cada módulo de comportamiento tiene una acción asociada, recibe la percepción directamente y comprueba una condición. Si esta se cumple, el módulo devuelve la acción a realizar.
- Un módulo se puede subsumir en otro. Si el módulo superior del esquema se cumple, se ejecuta este en lugar de los módulos inferiores.

Arquitectura de subsunción

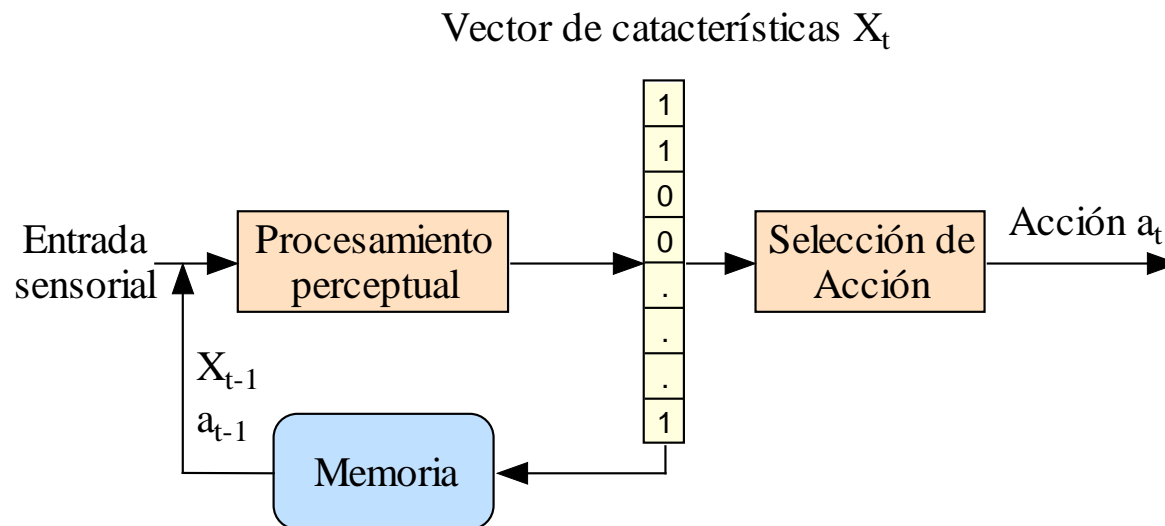


Agentes reactivos con memoria

- Limitaciones del sistema sensorial de un agente.
- Mejorar la precisión teniendo en cuenta la historia sensorial previa: sistemas con memoria

Agentes reactivos con memoria

la representación de un estado en el instante $t+1$ es función de la entradas sensoriales en el instante $t+1$, la representación del estado en el instante anterior t y la acción seleccionada en el instante anterior t .



Ejemplo

- Usaremos las características $w_i = s_i$ $i=2,4,6,8$ y las características restantes del siguiente modo

$w_1=1$ si en el instante anterior $w_2=1$ y el robot se movió al este
 $w_3=1$ si en el instante anterior $w_4=1$ y el robot se movió al sur
 $w_5=1$ si en el instante anterior $w_6=1$ y el robot se movió al oeste
 $w_7=1$ si en el instante anterior $w_8=1$ y el robot se movió al norte

$w_2\overline{w_4} \rightarrow$ este

$w_4\overline{w_6} \rightarrow$ sur

$w_6\overline{w_8} \rightarrow$ oeste

$w_8\overline{w_2} \rightarrow$ norte

$w_1 \rightarrow$ norte

$w_3 \rightarrow$ este

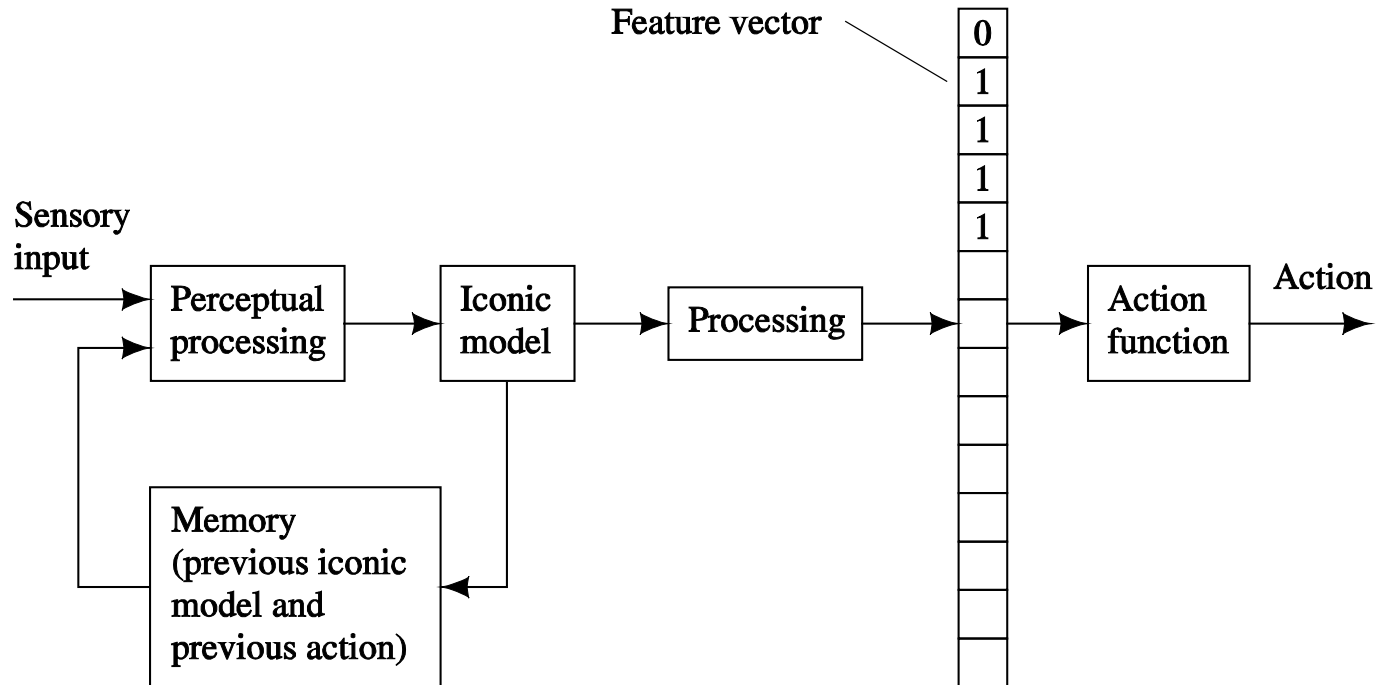
$w_5 \rightarrow$ sur

$w_7 \rightarrow$ oeste

$1 \rightarrow$ norte

Implementación de la memoria con representaciones icónicas

- Adicionalmente el robot podría utilizar otras estructuras de datos: matriz que almacene el mapa con las casillas libres u ocupadas en el momento en el que se percibieron.



© 1998 Morgan Kaufman Publishers

Campo de potencial artificial

	1	1	1	1	1	1	1	?
1	0	0	0	0	0	0	0	?
1	0	0	0	0	0	0	0	?
1	0	0	0	0	0	0	0	?
1	0	0	0	0	0	0	0	?
1	0	0	R	0	0	0	0	?
1	0	0	0	0	0	0	0	?
1	0	0	0	0	0	0	0	?
1	0	0	0	0	0	0	0	?
1	?	?	?	?	?	?	?	?
?	?	?	?	?	?	?	?	?

Componente atractiva:

$$p_a(X) = k_1 d(X)^2$$

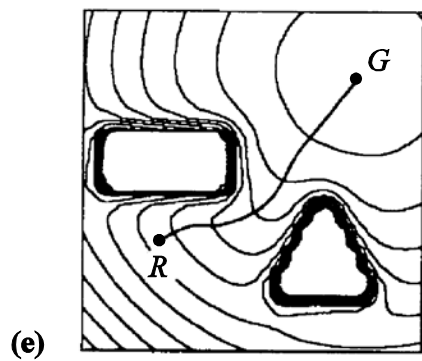
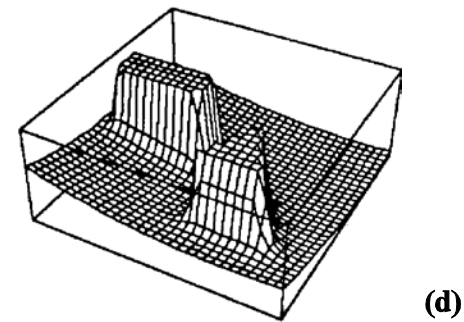
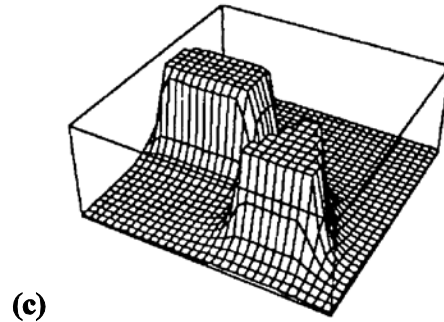
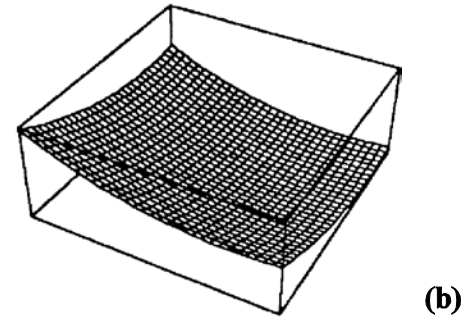
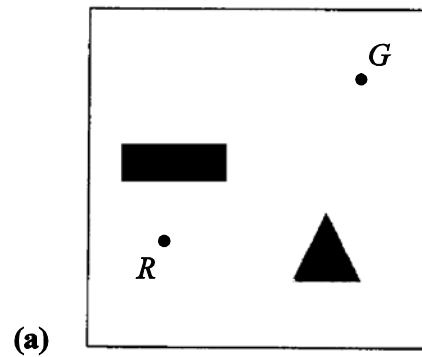
Componente repulsiva:

$$p_r(X) = \frac{k_2}{d_0(X)^2}$$

Potencial:

$$\text{Potencial} = p_a + p_r$$

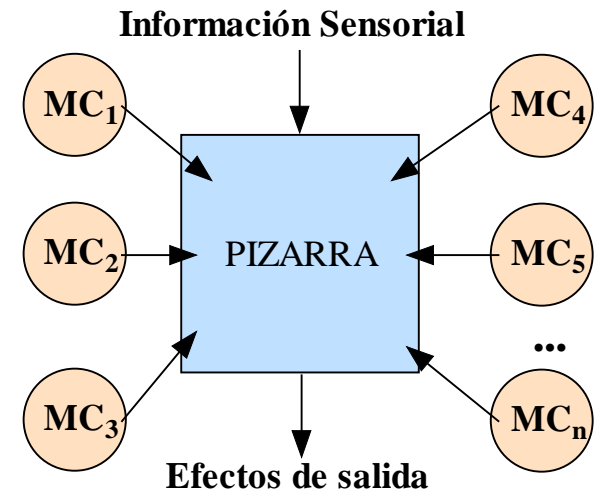
© 1998 Morgan Kaufman Publishers



© 1998 Morgan Kaufmann Publishers

Implementación de la memoria con sistemas basados en pizarras

- Son extensiones de los sistemas de producción.
- En el agente existen varios programas denominados **Módulos de Conocimiento (MC)**, formados por una parte de condición y otra parte de acción.
- Existe una memoria común a todos los MC denominada **pizarra**.

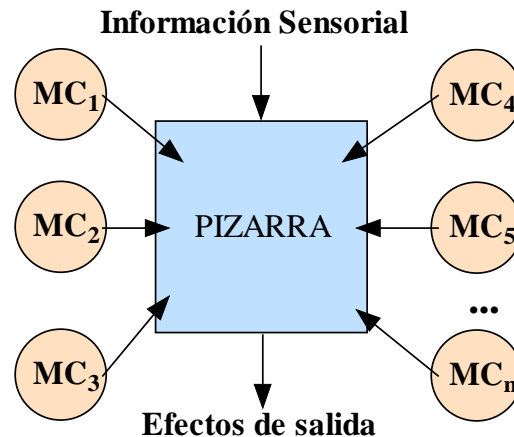


Implementación de la memoria con sistemas basados en pizarras

- Cada MC es “experto” en una parte concreta del problema a resolver.
- Cuando se cumple su condición, un MC puede actualizar la pizarra, realizar una acción concreta o ambas.
- Es necesario implementar un programa de **resolución de conflictos** cuando dos MCs pueden actuar simultáneamente, decidiendo cuál actúa y cuál no o, en su caso, el orden de ejecución de ambos.

Implementación de la memoria con sistemas basados en pizarras

- La actualización de una parte de la pizarra correspondiente a un MC puede desencadenar la ejecución de otros MCs.
- La pizarra, por tanto, alberga la solución que se está construyendo conforme al objetivo general del agente.



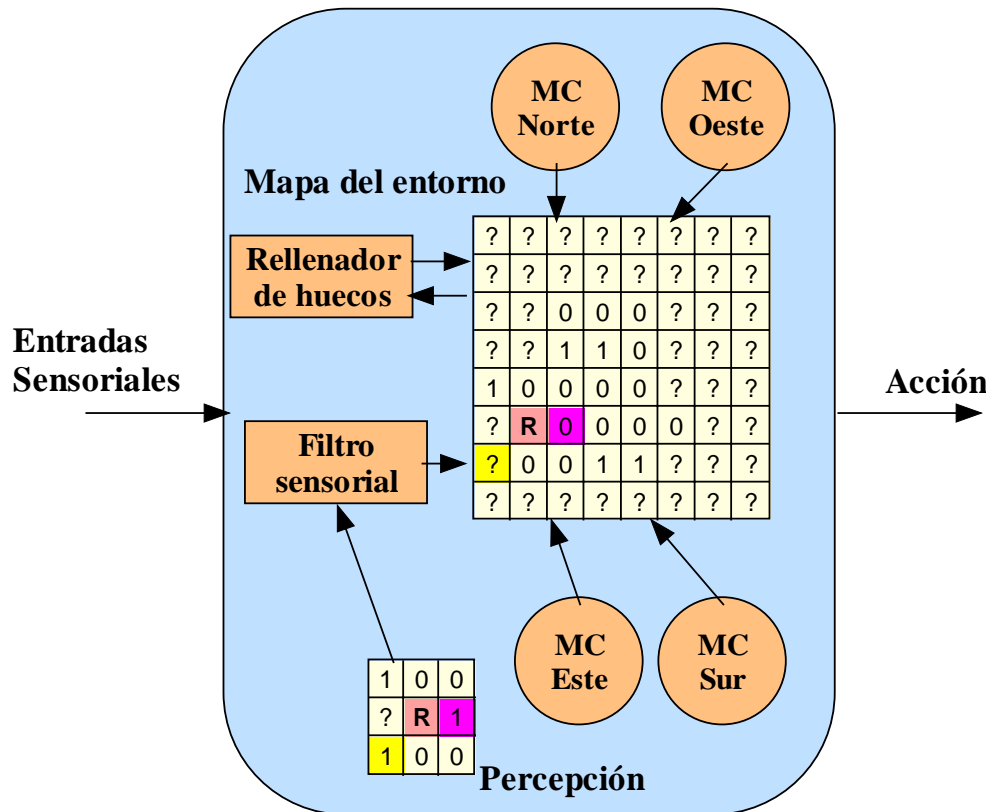
Implementación de la memoria con sistemas basados en pizarras

Ejemplo: Robot para salir de un laberinto.

- La pizarra contiene la información leída desde los sensores (que puede ser imperfecta debido a errores de los mismos). También contiene un mapa del laberinto, que puede tener errores debido a previas lecturas erróneas de los sensores, junto con la posición del robot en el mapa.
- Contamos con 4 módulos MC de acción de movimiento (“Norte”, “Sur”, “Este”, “Oeste”).
- Contamos con 2 MC adicionales:
 - Rellenador de huecos para rellenar el mapa del laberinto.
 - Filtro sensorial para arreglar errores en el mapa.

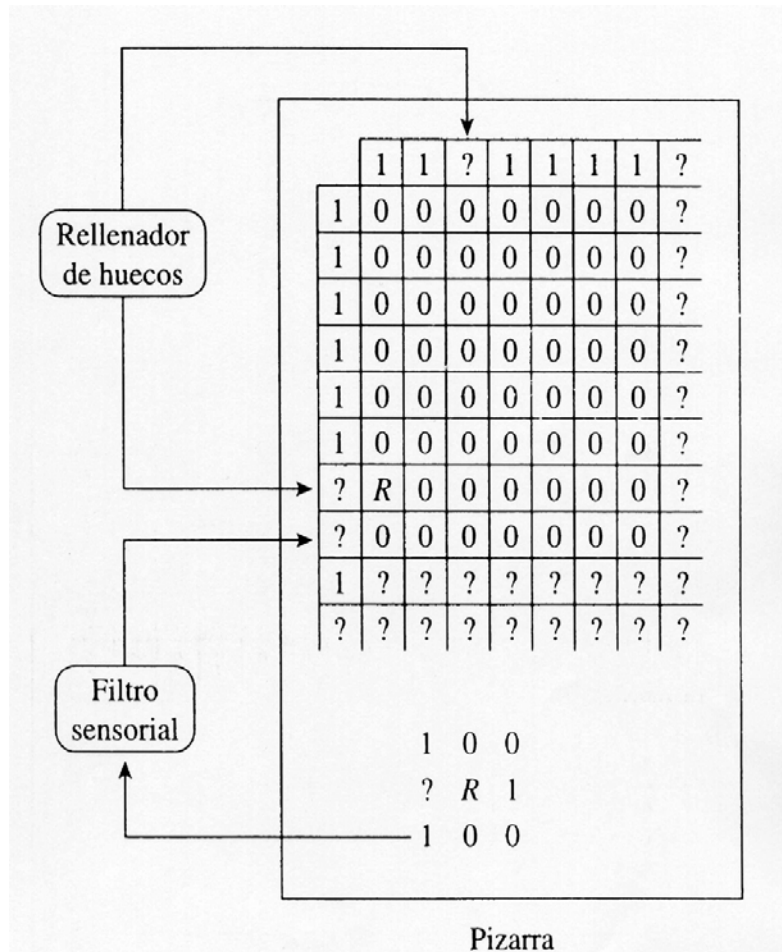
Agentes reactivos con memoria

- Implementación de la memoria con sistemas basados en pizarras. Ejemplo: Agente en el mundo cuadrulado.



- En el ejemplo, el filtro sensorial detecta un error entre la percepción y el estado del mundo.
- El rellенador de huecos usa la información sensorial para rellenar las nuevas casillas desconocidas por el robot.

Implementación de la memoria con sistemas basados en pizarras. Ejemplo: Agente en el mundo cuadrulado

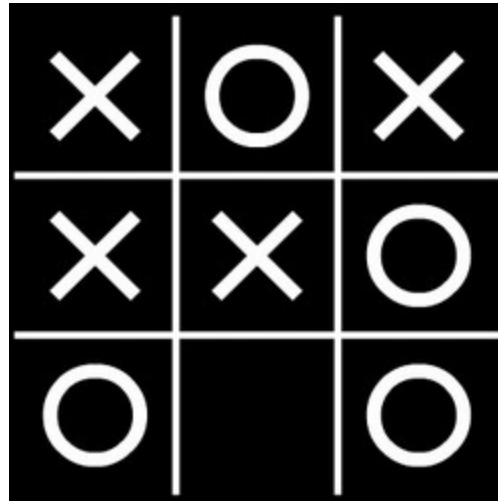


- En el ejemplo, el filtro sensorial detecta un error entre la percepción y el estado del mundo.
- El rellenedor de huecos usa la información sensorial para rellenar las nuevas casillas desconocidas por el robot.

Ejemplo de agente reactivo: un robot que recorre un pasillo



Ejemplos de agente reactivo: un agente que juega al tres en raya



Características de los agentes reactivos

- Se diseñan completamente y por tanto es necesario anticipar todas las posibles reacciones para todas las situaciones
 - Realizan pocos cálculos
 - Almacenan todo en memoria

Diseño de un agente deliberativo

- El agente dispone de un modelo del mundo en el que habita,
- El agente dispone de un modelo de los efectos de sus acciones sobre el mundo,
- El agente es capaz de razonar sobre esos modelos para decidir que hacer para conseguir un objetivo.

El espacio de estados

- Modelo del mundo/Estado del agente: representación
- Espacio de estados
- Operadores de cambio de estado: acciones del agente.
- Acciones del agente: representación

La búsqueda en un espacio de estados

Problema:

- El agente se encuentra en un **estado inicial (origen, de partida, etc)**,
- Se desea alcanzar un **estado final (meta, objetivo, etc.)**

Búsqueda en el espacio de estados – Resolución del problema mediante análisis de las distintas acciones

La búsqueda en un espacio de estados

- A la secuencia de acciones que lleva al agente desde un **estado inicial** hasta un **estado destino** se denomina **plan**.
- La búsqueda de dicha secuencia se denomina **planificación**.

La búsqueda en un espacio de estados

Representación en forma de grafo (**grafo de estados**) :

- Nodo: estado = estructura de datos
- Arco: asociado a una acción. Une un nodo (estado) con el nodo (estado) resultante de la acción.
- Búsqueda en espacio de estados: búsqueda en grafos

Búsqueda en Grafos

- **Grafos explícitos:** nodos y arcos que unen dichos nodos.
- **Grafos implícitos:** reglas de representación de estados y reglas de cambio de estado por medio de operadores.

Búsqueda en espacios de estados

- **“Pequeño”**: se puede representar la totalidad del espacio de estados con un grafo explícito y puede buscarse un camino que nos lleve desde el estado inicial hasta el estado objetivo.
- **“Grande”** : se va generando un grafo explícito según se va resolviendo el problema en cada paso. Técnicas de búsqueda.

Grafo del Espacio vs. Grafo de la búsqueda (árbol)

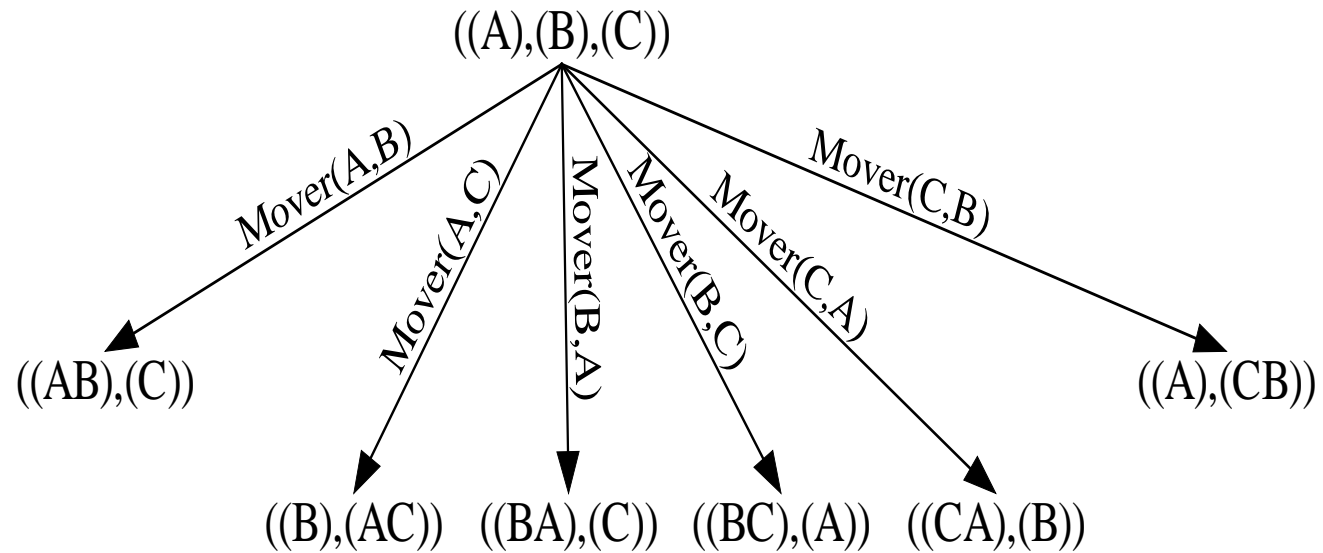
“Pequeño”: El mundo de bloques

- Supongamos un mundo compuesto por una mesa sobre la que hay tres bloques: **A, B, C**.
- En cada momento, se conoce el estado del sistema. Lo modelamos como una lista de listas de (objetos sobre objetos): ((A),(BC)); ((AB),(C))

El mundo de los bloques

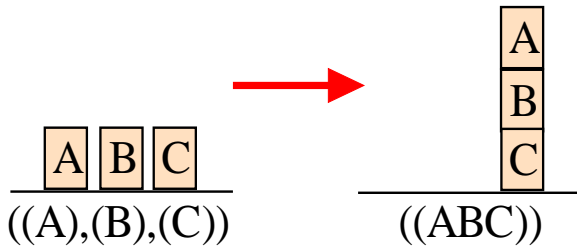
- En cada momento, se dispone de la operación **mover(x,y)** para poner **x** sobre **y**, donde **x**=**{A, B, C}** e **y**=**{A, B, C, Suelo}**.
- Asumimos que se descartan los **operadores imposibles mover(A,A), mover(B,B), mover(C,C), etc.**, para cada estado.

El mundo de los bloques

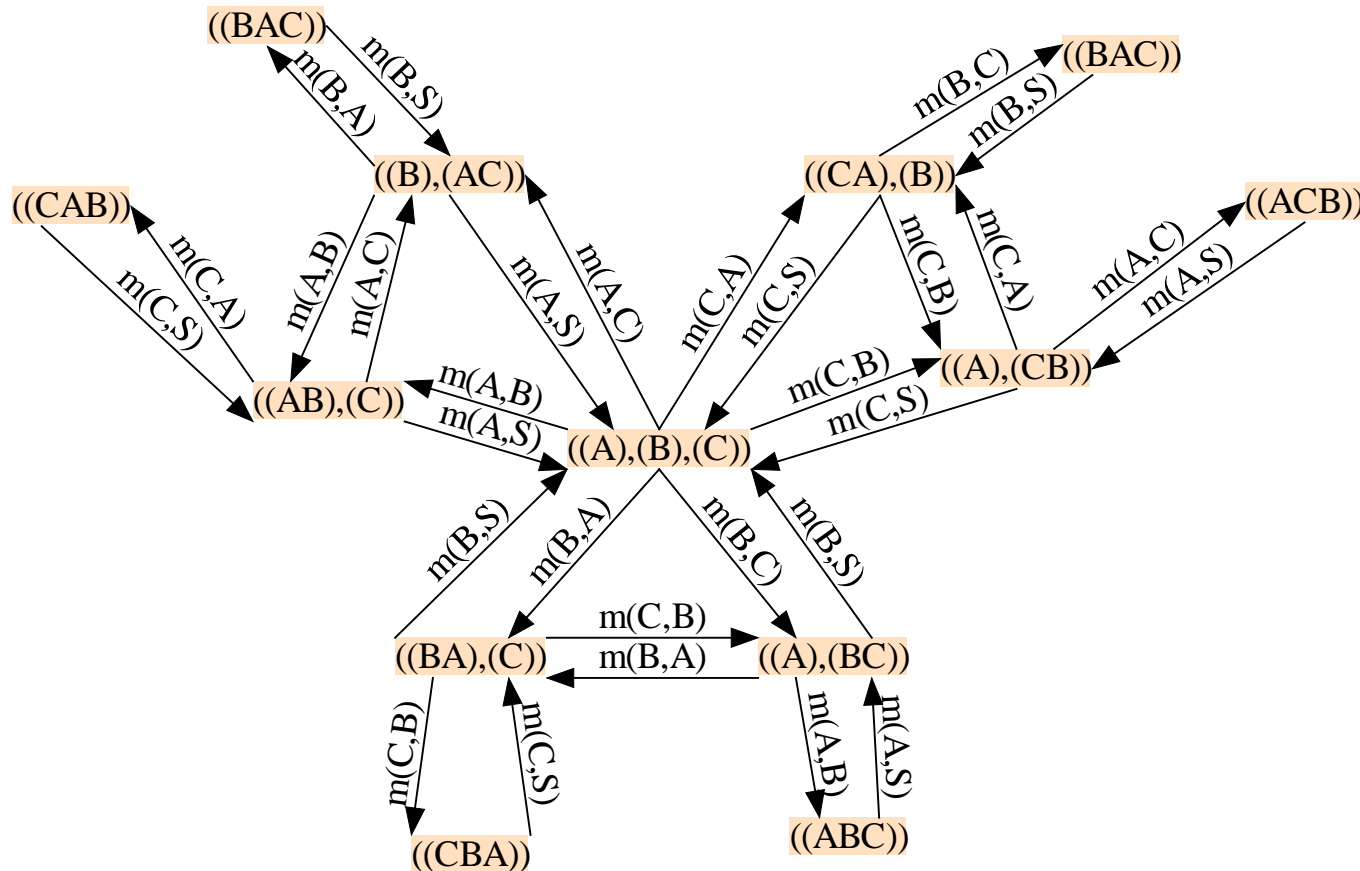


El mundo de bloques

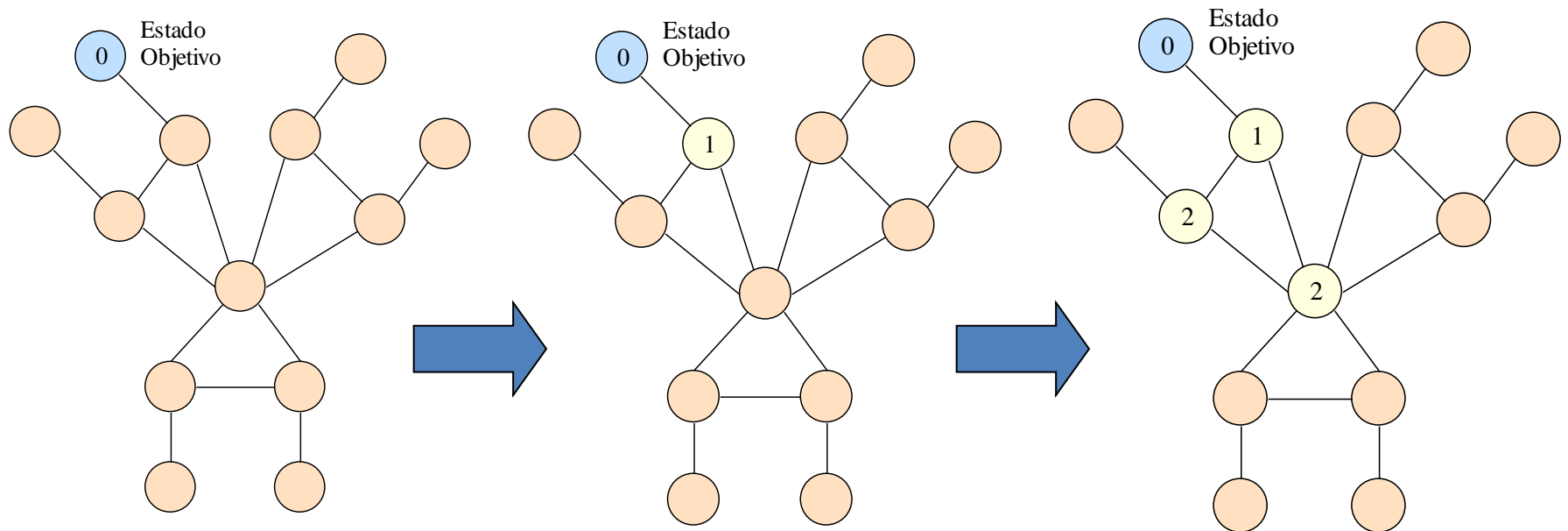
- Estado inicial: todos los bloques están directamente sobre la mesa:
 $((A), (B), (C))$
- Estado objetivo: **A** quede sobre **B**, **B** quede sobre **C**, y **C** esté en la mesa: **$((ABC))$** .



Espacio (Grafo) de estados en el mundo de bloques



Búsqueda



Búsqueda

- **Ejemplo de planificación en el mundo de los bloques:**
Planificación de acciones.

Estado Inicial: ((ABC))

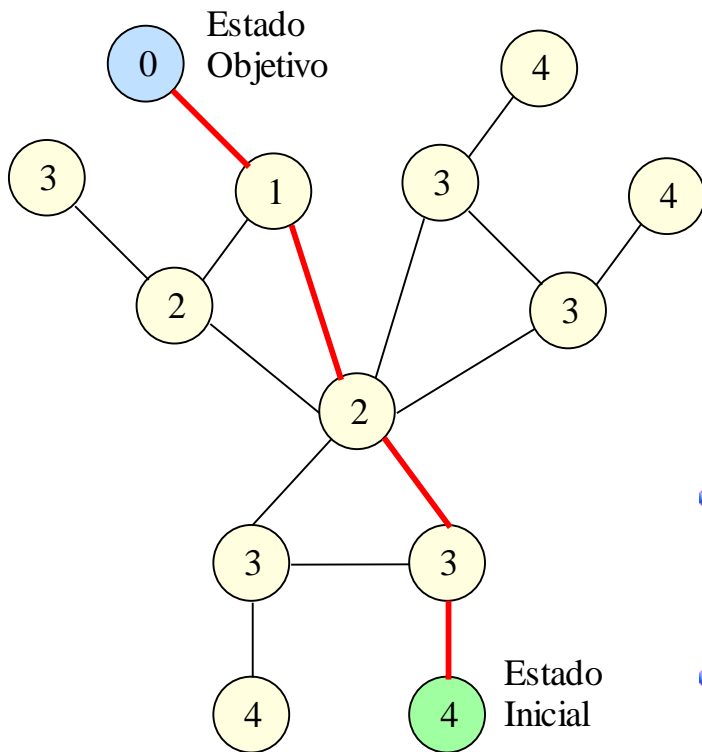
Acción 1: **Mover(A, Suelo)**

Acción 2: **Mover(B, Suelo)**

Acción 3: **Mover(A, C)**

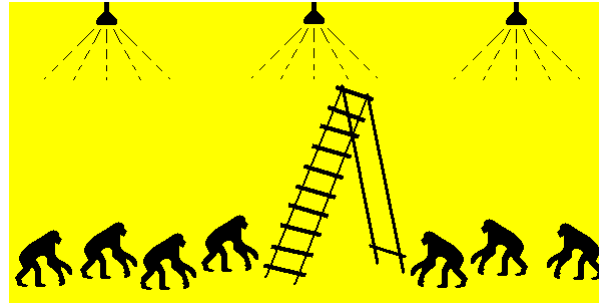
Acción 4: **Mover(B, A)**

Estado objetivo alcanzado: ((BAC))



- **Propagación de movimientos:**
Recorrido en anchura.
- **Búsqueda del plan:** Recorrido en profundidad.

Ejemplo de agente deliberativo: Problema del mono y los plátanos

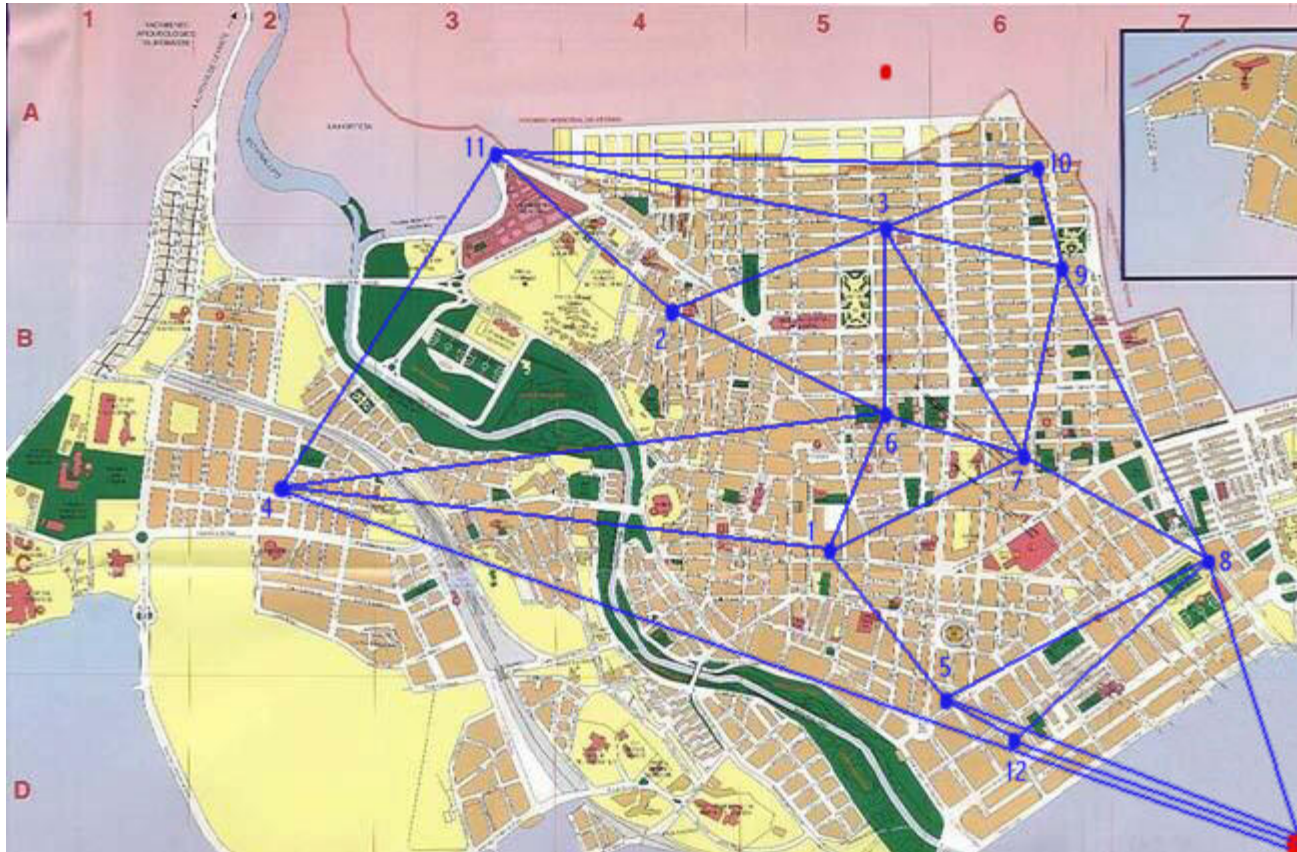


- “Un mono está en la puerta de una habitación. En el centro de la habitación hay un plátano colgado del techo, pero no puede alcanzarlo desde el suelo. En la ventana de la habitación hay una caja, que el mono puede mover y a la que puede encaramarse para alcanzar el plátano. El mono puede realizar las siguientes acciones: desplazarse de la puerta al centro, del centro a la ventana y viceversa; empujar la caja a la vez que se desplaza; subirse y bajarse de la caja; coger el plátano. El problema consiste en encontrar una secuencia de acciones que permita al mono coger el plátano.”

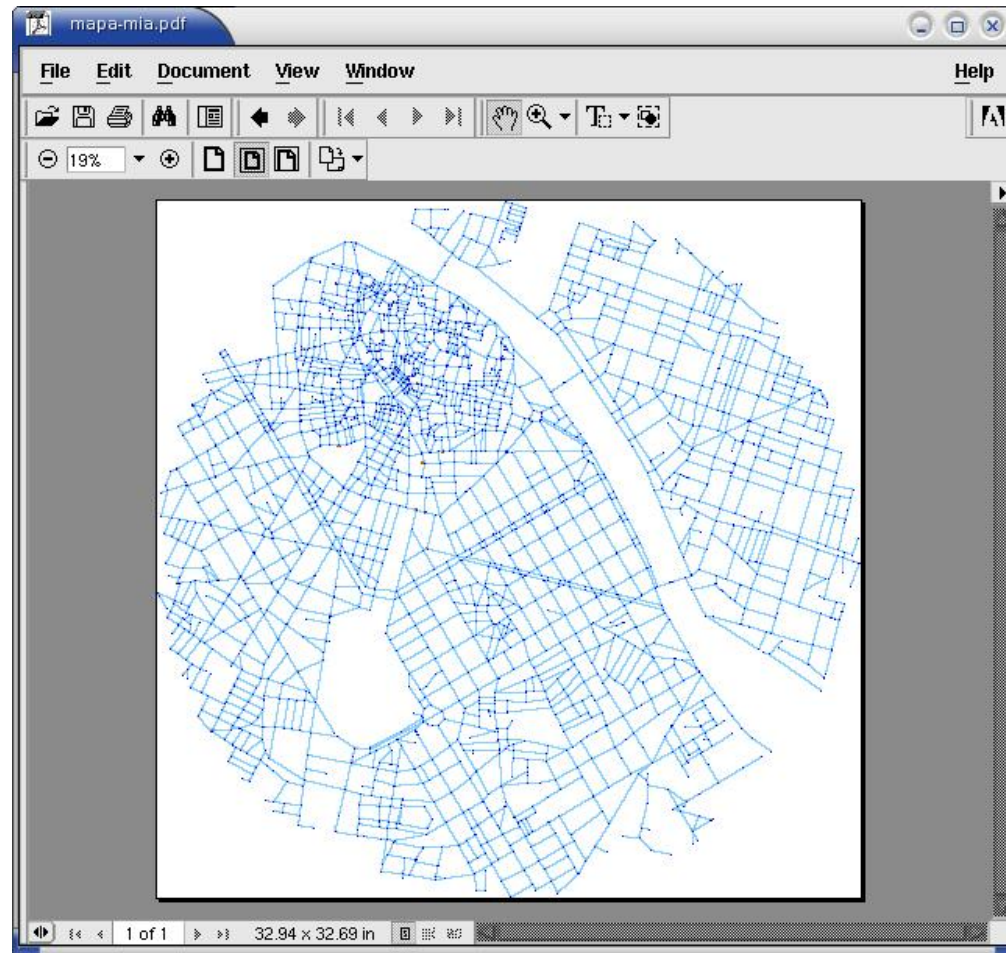
Problema del mono y los plátanos

- Como estado se puede utilizar una lista con cuatro elementos (X, Y, W, Z) donde:
 - X: posición del mono en la habitación (puerta, centro, ventana).
 - Y: situación del mono respecto a la caja (suelo, caja).
 - W: posición de la caja en la habitación (puerta, centro, ventana).
 - Z: posesión del plátano (tiene, no_tiene).
- Para describir todas las posibles acciones del mono, se necesitan seis operadores: andar, empujar, subir, bajar, coger y soltar. Sin embargo, para el problema que nos ocupa, son suficientes cuatro operadores: andar, empujar, subir y coger; nos limitaremos a estos 4 operadores .
- El operador andar se puede definir como:
 - (X, suelo, W, Z) -----> (Y, suelo, W, Z)
- para indicar que el mono se desplaza de la posición X a la posición Y.
- El estado se puede representar por la estructura *estado(X, Y, W, Z)*, que simplemente refleja la definición del estado.

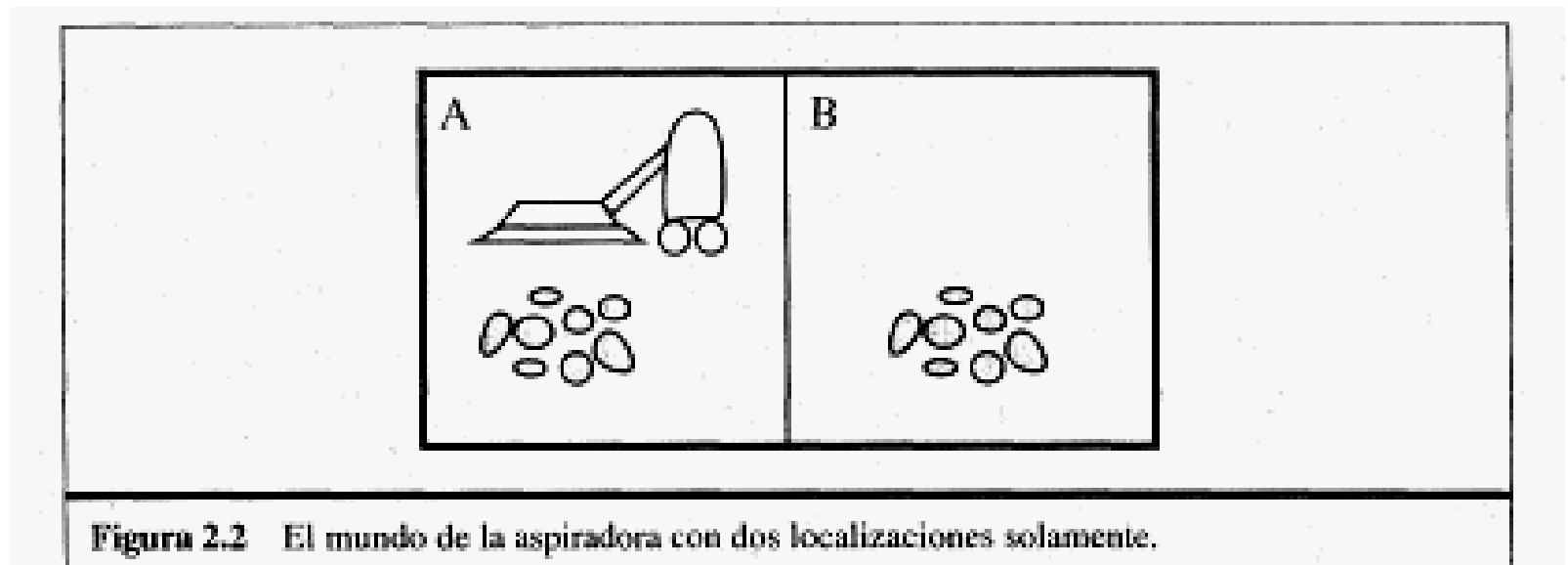
Ejemplo de agente deliberativo: Problema del viajante de comercio



Ejemplo de agente deliberativo: Mapa de Carreteras

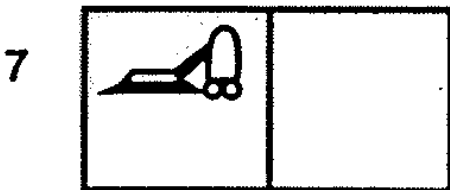
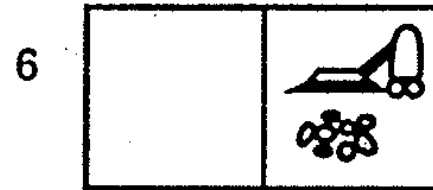
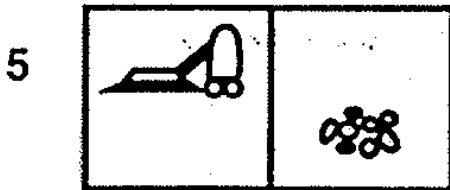
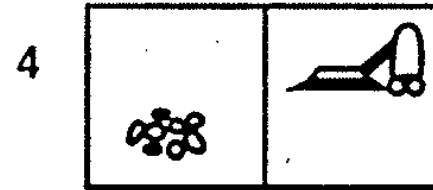
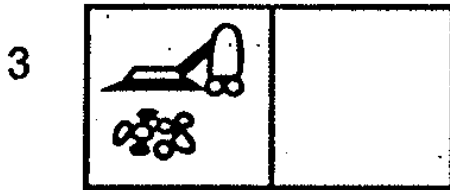
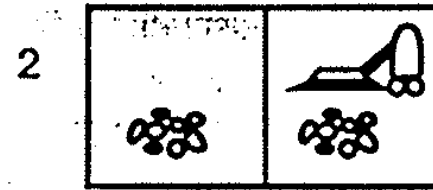
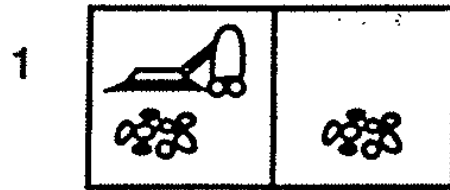


Ejemplo de un agente deliberativo: Problema de la aspiradora



- **Estados:** el agente está en una de dos localizaciones, cada una de las cuales puede o no contener suciedad. Así, hay $2 \times 2^2 = 8$ posibles estados del mundo.
- **Estado inicial:** cualquier estado puede designarse como un estado inicial.
- **Función sucesor:** ésta genera los estados legales que resultan al intentar las tres acciones (*Izquierda*, *Derecha* y *Aspirar*). En la Figura 3.3 se muestra el espacio de estados completo.
- **Test objetivo:** comprueba si todos los cuadrados están limpios.
- **Costo del camino:** cada costo individual es 1, así que el costo del camino es el número de pasos que lo compone.

Problema de la aspiradora



Problema de la aspiradora

Russell Norvig

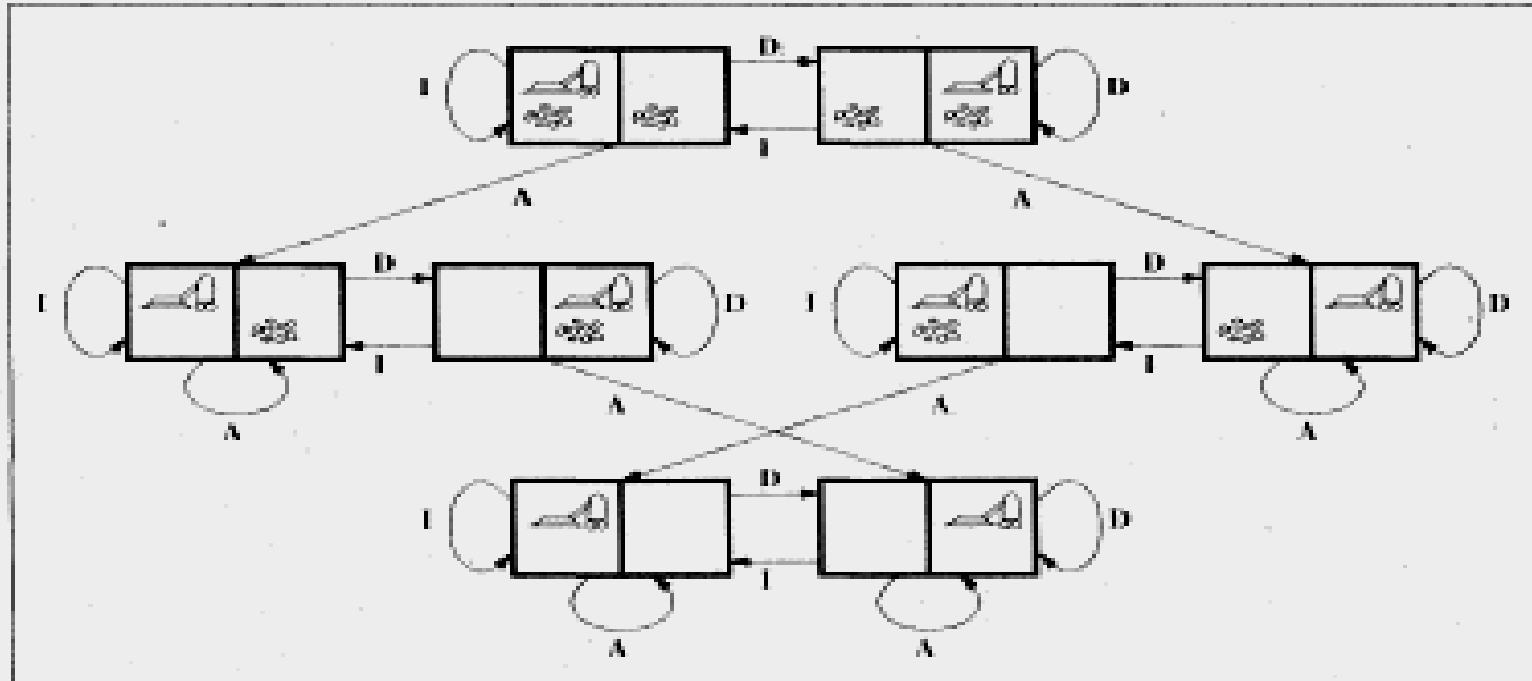
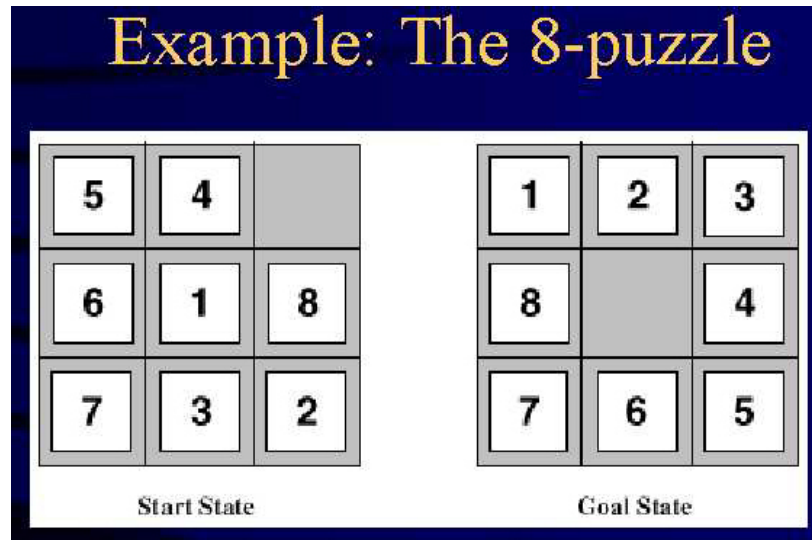


Figura 3.3 Espacio de estados para el mundo de la aspiradora. Los arcos denotan las acciones: I = *Izquierda*, D = *Derecha*, A = *Aspirar*.

Espacio de estados Grande



Espacio de estados Grande

Estado: configuración del tablero

- una matriz 3x3 con entradas del 0 al 8
- un vector de 9 términos de 0 a 8.

Movimientos:

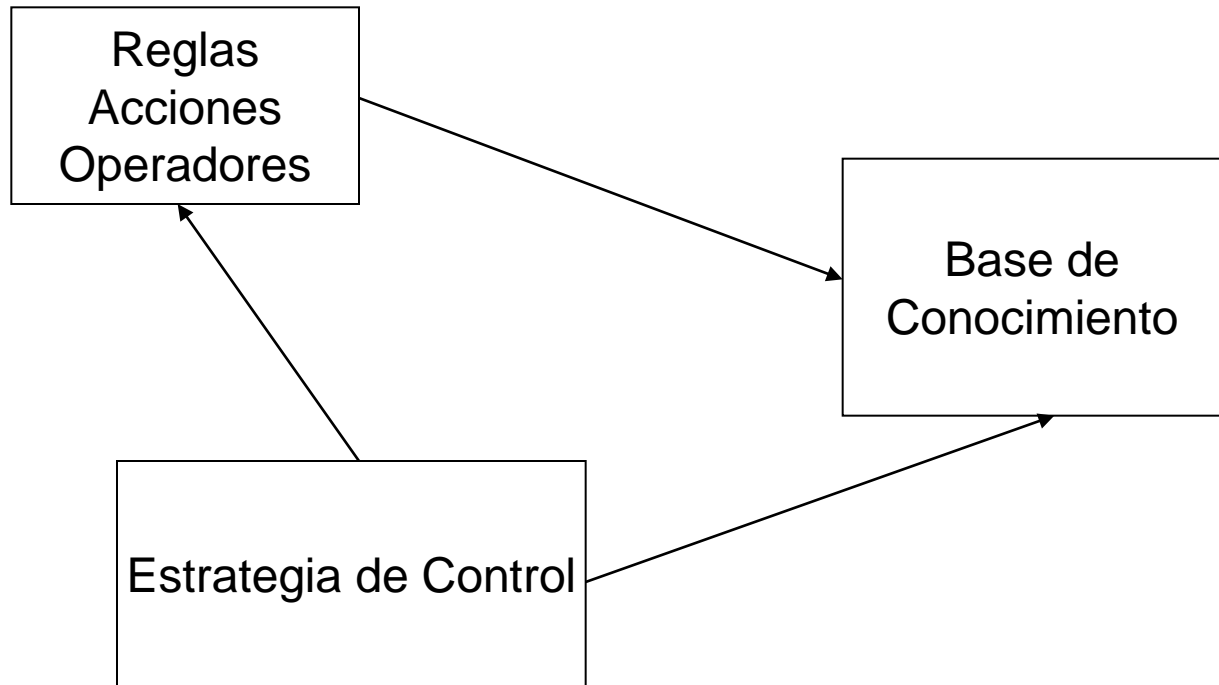
- Mover casillas adyacentes al hueco
- Mover hueco a casillas adyacentes (4 pos. Max.)

Espacio de estados Grande

**Numero de estados: $9!$, en realidad $9!/2$
 $9!=362.880$**

- Imposible representar explícitamente todo el grafo de estados.
- Necesidad de recurrir al grafo implícito y explicitar solo el grafo o árbol de búsqueda.
- Necesidad de técnicas de búsqueda.

Técnicas de Búsqueda



Estrategias de control (Russell-Norvig)

- Escoger el nodo a expandir,
 - Escoger como expandirlo.
-
- Debe producir cambios
 - Debe ser sistemática
 - Debe “garantizar” construir la solución

Procedimiento Búsqueda

1. DATOS \leftarrow base de datos inicial
2. **until** DATOS satisface la condición de terminación
do
3. **begin**
4. **select** alguna regla R en el conjunto de reglas que pueda ser aplicada a DATOS
5. DATOS \leftarrow resultado de aplicar R a DATOS
6. **end**

Estrategias de control (Russell-Norvig)

función *BÚSQUEDA-ÁRBOLES*(*problema*, *frontera*) **devuelve** una solución o fallo

frontera \leftarrow *INSERTA*(*HACER-NODO*(*ESTADO-INICIAL*[*problema*]), *frontera*)

hacer bucle

si *VACIA?*(*frontera*) **entonces devolver** fallo.

nodo \leftarrow *BORRAR-PRIMERO*(*frontera*)

si *TEST-OBJETIVO*[*problema*] aplicado al *ESTADO*[*nodo*] es cierto

entonces devolver *SOLUCIÓN*(*nodo*).

frontera \leftarrow *INSERTAR-TODO*(*EXPANDIR*(*nodo*, *problema*), *frontera*)

Estrategias de control(Russell-Norvig

función EXPANDIR(*nodo,problema*) **devuelve** un conjunto de nodos

sucesores \leftarrow conjunto vacío

para cada (*acción,resultado*) **en** SUCESOR-FN[*problema*](ESTADO[*nodo*]) **hacer**

s \leftarrow un nuevo NODO

ESTADO[*s*] \leftarrow *resultado*

NODO-PADRE[*s*] \leftarrow *nodo*

ACCIÓN[*s*] \leftarrow *acción*

COSTO-CAMINO[*s*] \leftarrow COSTO-CAMINO[*nodo*] + COSTO-INDIVIDUAL(*nodo,acción,s*)

PROFUNDIDAD[*s*] \leftarrow PROFUNDIDAD[*nodo*] + 1

añadir *s* a *sucesores*

devolver *sucesores*

Rendimiento de una estrategia de Control

- **Complejidad:** ¿encuentra una solución(si existe)?
- **Optimalidad:** ¿Encuentra una solución optima en algun sentido?
- **Complejidad en tiempo,**
- **Complejidad en espacio, irrevocable vs tentativa**
- **Costo de la búsqueda, Costo total.**

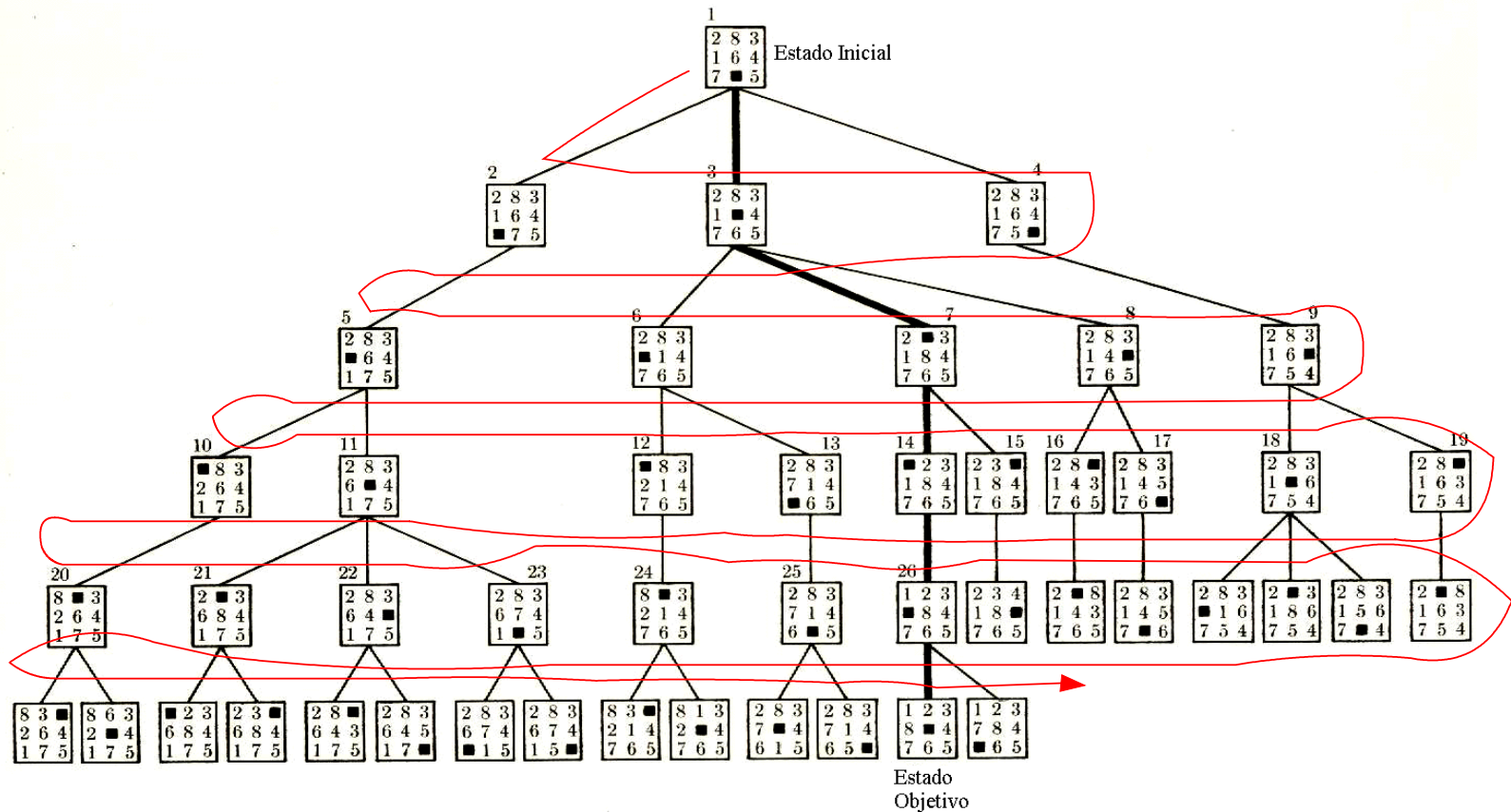
Búsqueda sin información

- Búsqueda en anchura,
- Búsqueda con costo uniforme,
- Búsqueda en profundidad,
- Búsqueda bidireccional.

Búsqueda en anchura

Desde el estado inicial analizar todos los sucesores de cada nodo antes de pasar al nivel siguiente en el árbol de búsqueda

Búsqueda en anchura



Búsqueda en anchura

1. Crear una variable llamada LISTA-NODOS y asignarle el estado inicial
2. Hasta que se encuentre un estado objetivo o LISTA-NODOS esté vacía, hacer:
 1. Eliminar el primer elemento de LISTA-NODOS y llamarlo E. Si LISTA-NODOS está vacía, terminar
 2. Para cada regla que se empareje con el estado descrito en E hacer:
 1. Aplicar la regla para generar un nuevo estado
 2. Si el nuevo estado es un estado objetivo, terminar y devolver este estado
 3. En caso contrario, añadir el nuevo estado al final de LISTA-NODO

Búsqueda con costo

Se supone que el expandir el nodo E_i con la regla R_{ij} para obtener el nodo E_j tiene un costo C_{ij} .

De este modo cualquier nodo E_k en el árbol de búsqueda tiene un costo C_k igual a la suma de los costos de los arcos que conducen a él desde la raíz del árbol.

El procedimiento de búsqueda expande el nodo E_p con menor C_p de todos los de la frontera.

Búsqueda con costo

1. Poner el nodo inicial s en una lista, llamada ABIERTOS, de nodos no expandidos.
2. Crear una lista, llamada CERRADOS, de nodos expandidos, inicialmente vacía.
3. Si ABIERTOS está vacía no existe solución. Terminar.
4. Suprimir de ABIERTOS el nodo i con mínima $g(i)$ y colocarlo en CERRADOS.
5. Si i es un nodo objetivo se ha encontrado la solución. Terminar.
6. En otro caso, expandir el nodo i , si no tiene sucesores ir a 3.
7. Para cada nodo sucesor j del nodo i , insertarlo correctamente en ABIERTOS, asignando $g(j)=g(i)+c(i,j)$.
8. Ir a 3.

Búsqueda con costo

Esta estrategia se reduce a la búsqueda en anchura si $C_{ij}=1$ para todo i y todo j .

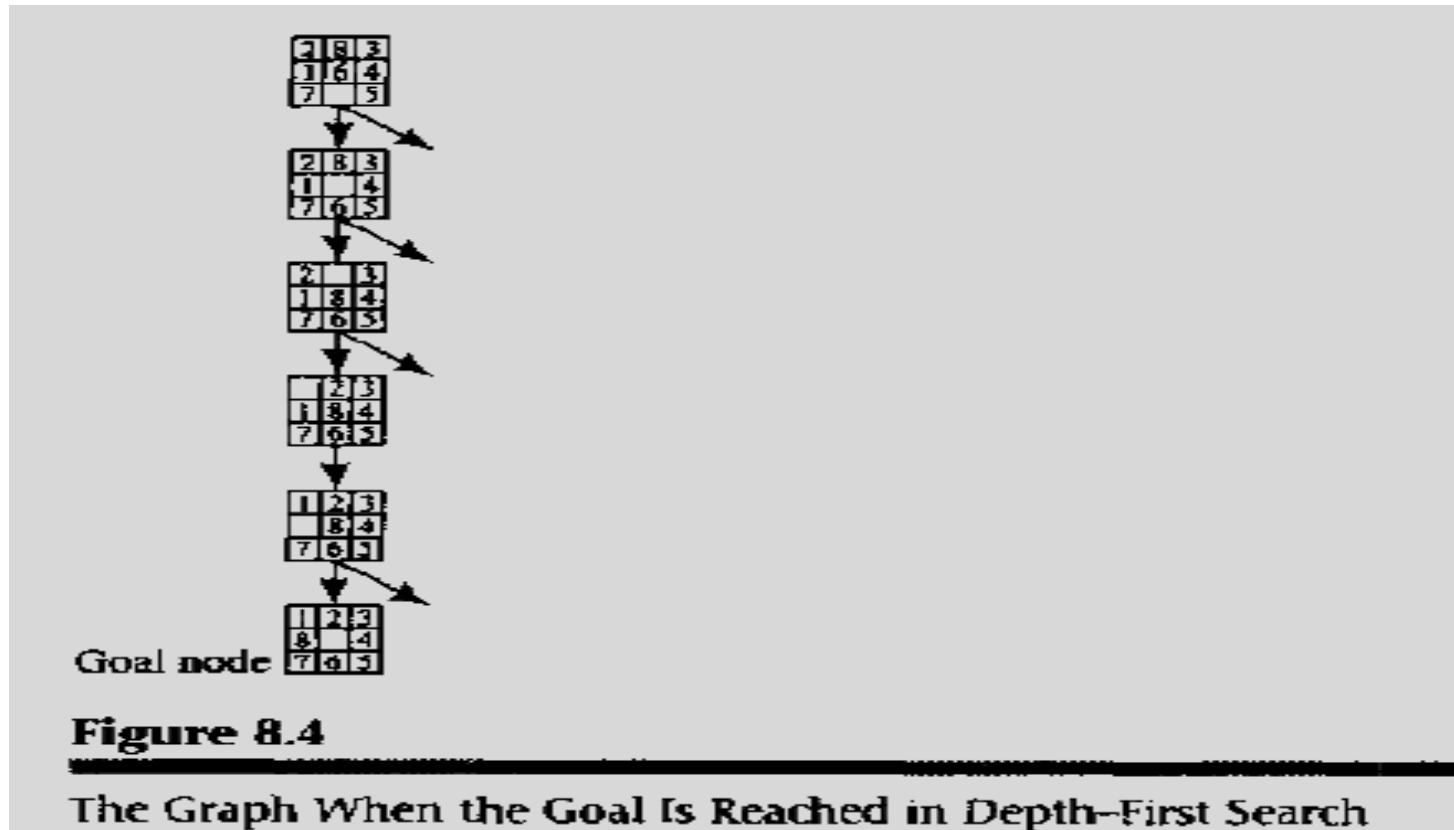
El método es completo y optimal en costo si C_{ij} es mayor que una cierto C para todo i y todo j .

Se puede mejorar considerando el hijo de menor costo de todos los de la frontera (complica algorítmicamente)

Búsqueda primero en profundidad

- Desde el estado inicial ir analizando un sucesor del nodo de mayor nivel generado hasta el momento.

Búsqueda primero en profundidad



Búsqueda primero en profundidad

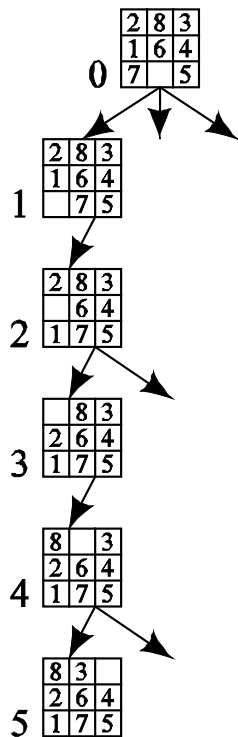
1. Si el estado objetivo es un estado final: STOP/ EXITO
2. Hasta encontrar un éxito o un fracaso hacer:
 - a) Generar un sucesor, **E**, del estado inicial que no haya sido analizado. Si no lo hay marcar FRACASO.
 - b) Llamar recursivamente a este algoritmo con **E** como estado inicial.
 - c) Alcanzado un ÉXITO o un FRACASO Stop.

Decisión ¿Qué guardar?

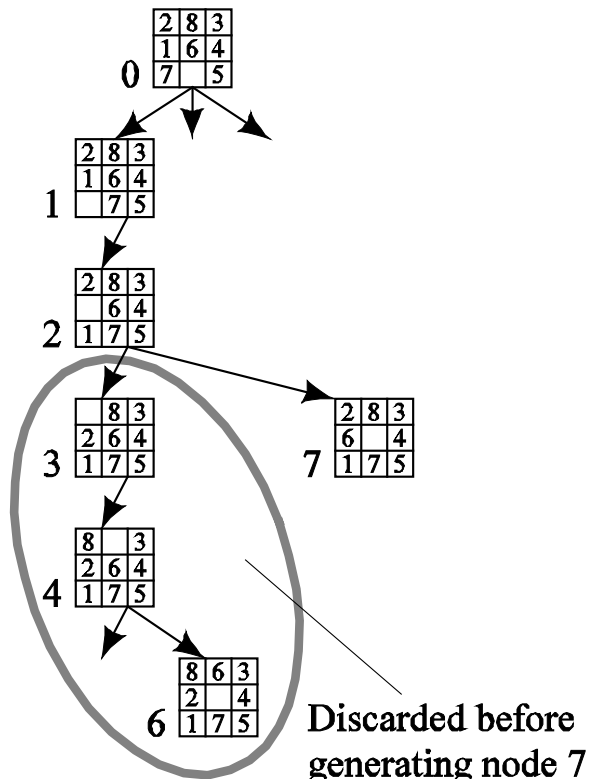
Búsqueda primero en profundidad

- Problema: profundizar indefinidamente
Solución: Acotar la profundidad,
 - Búsqueda con retroceso (Backtracking) a profundidad fijada o retroceso cronológico.
 - Búsqueda con profundización iterativa

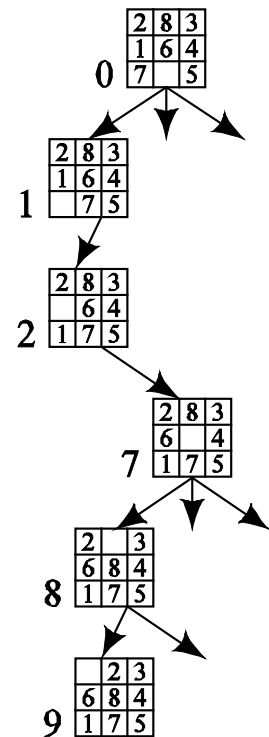
Búsqueda con backtracking cronológico



(a)



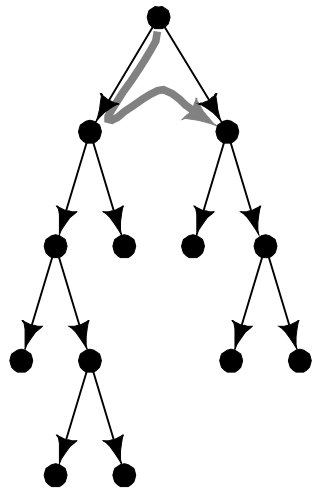
(b)



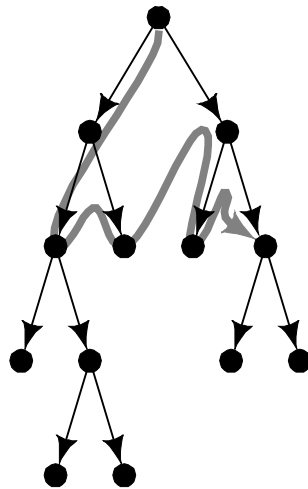
(c)

© 1998 Morgan Kaufman Publishers

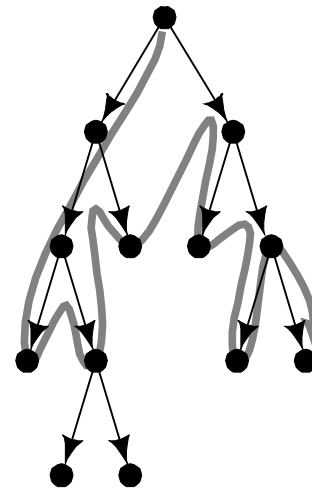
Descenso iterativo



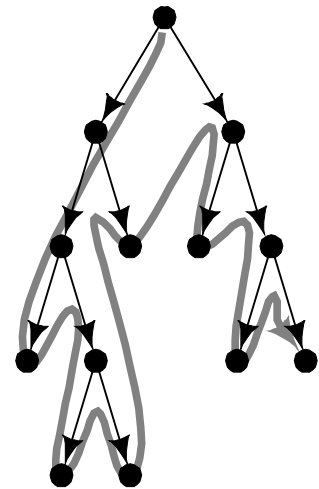
Depth bound = 1



Depth bound = 2



Depth bound = 3



Depth bound = 4

© 1998 Morgan Kaufman Publishers

Anchura /Profundidad

- La búsqueda en profundidad consume menos memoria,
- Con “un poco de suerte” la búsqueda en profundidad encuentra pronto el camino del nodo inicial al nodo final,
- La búsqueda en anchura no queda atrapada en callejones sin salida,
- La búsqueda en anchura encuentra siempre el camino, si existe, y además el más corto.

Búsqueda en grafos: Nodos repetidos

- Descripción para etiquetar el nodo inicial
- Las funciones para transformar las descripciones de los estados: operadores
- Una memoria para guardar los nodos previamente obtenidos
- Una condición de finalización.

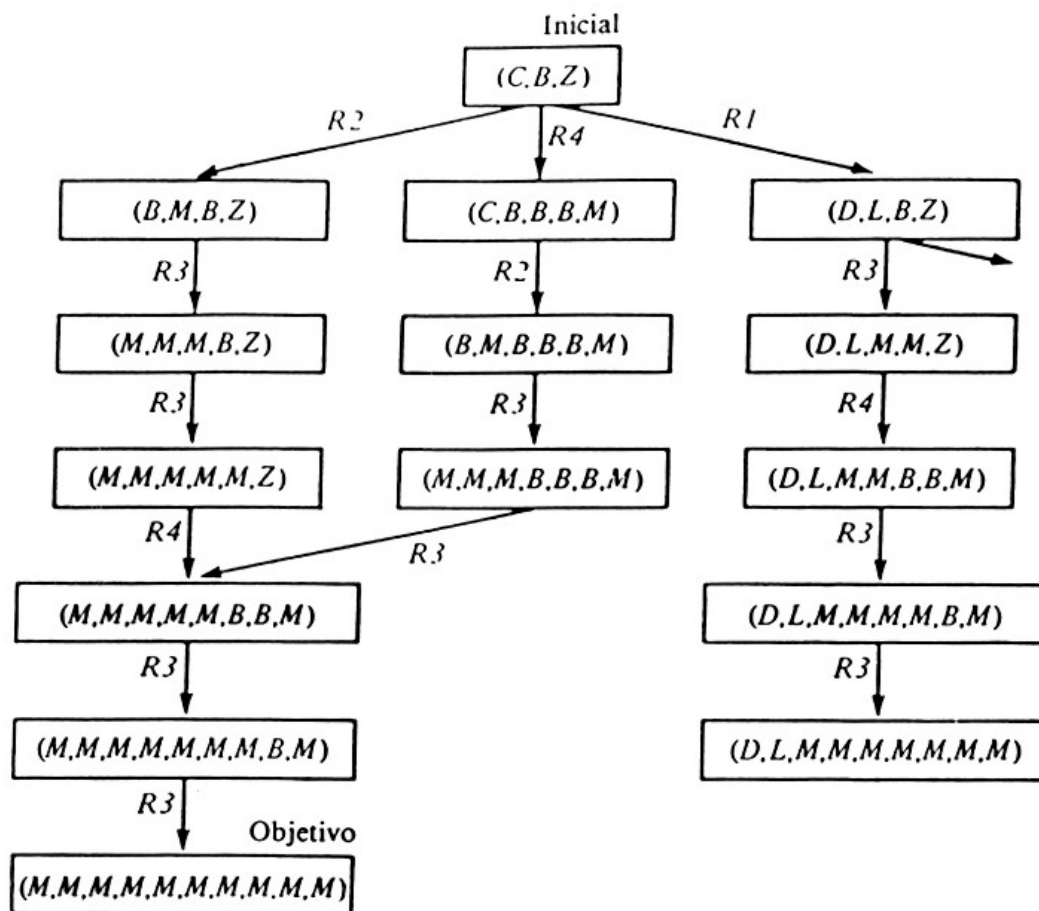
Problemas descomponibles

- Son aquellos en los que el problema se puede dividir en subproblemas que podemos resolver de forma independiente .
- La solución global se obtiene “integrando” las soluciones de los subproblemas.

Problemas descomponibles

- Base de datos inicial (C,B,Z)
- Operadores
 - R1: $C \longrightarrow (D,L)$
 - R2: $C \longrightarrow (B,M)$
 - R3: $B \longrightarrow (M,M)$
 - R4: $Z \longrightarrow (B,B,M)$
- Objetivo: (MMMMMMMMMMMM)

Resolución del problema

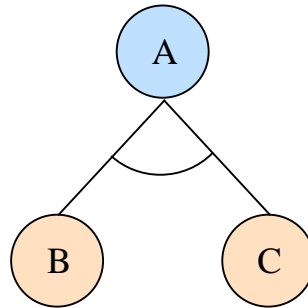


Grafo Y/O

- Descomposición de problemas: arcos Y
- Resolución de problemas: arcos O
- Concepto de solución: subgrafo solución

Grafo Y/O

- **Grafo Y:** Para completar el objetivo/tarea **A**, es necesario terminar antes los objetivos/tareas **B** y **C**.

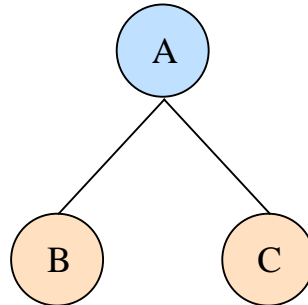


- En el cálculo proposicional, la expresión del grafo Y anterior correspondiente sería de la siguiente forma:

$$\mathbf{B \cdot C \rightarrow A}$$

Grafo Y/O

- **Grafo O:** Para completar el objetivo/tarea **A**, es necesario terminar antes o bien el objetivo/tarea **B**, o bien el objetivo/tarea **C**.

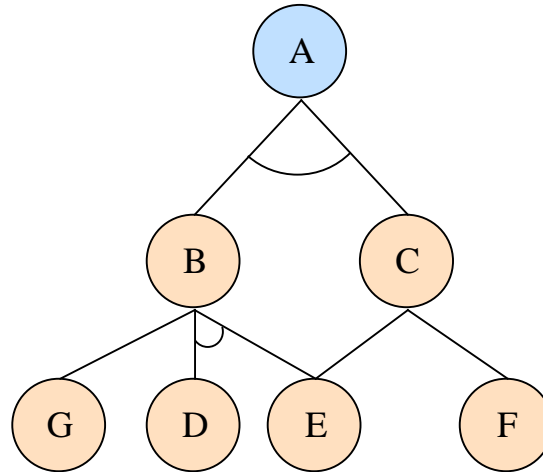


- En el cálculo proposicional, la expresión del grafo O anterior correspondiente sería de la siguiente forma:

$$\mathbf{B+C \rightarrow A}$$

Grafo Y/O

- **Grafo Y/O:** Combinación de grafos Y y grafos O que indican el orden de consecución de tareas a realizar para alcanzar el objetivo.



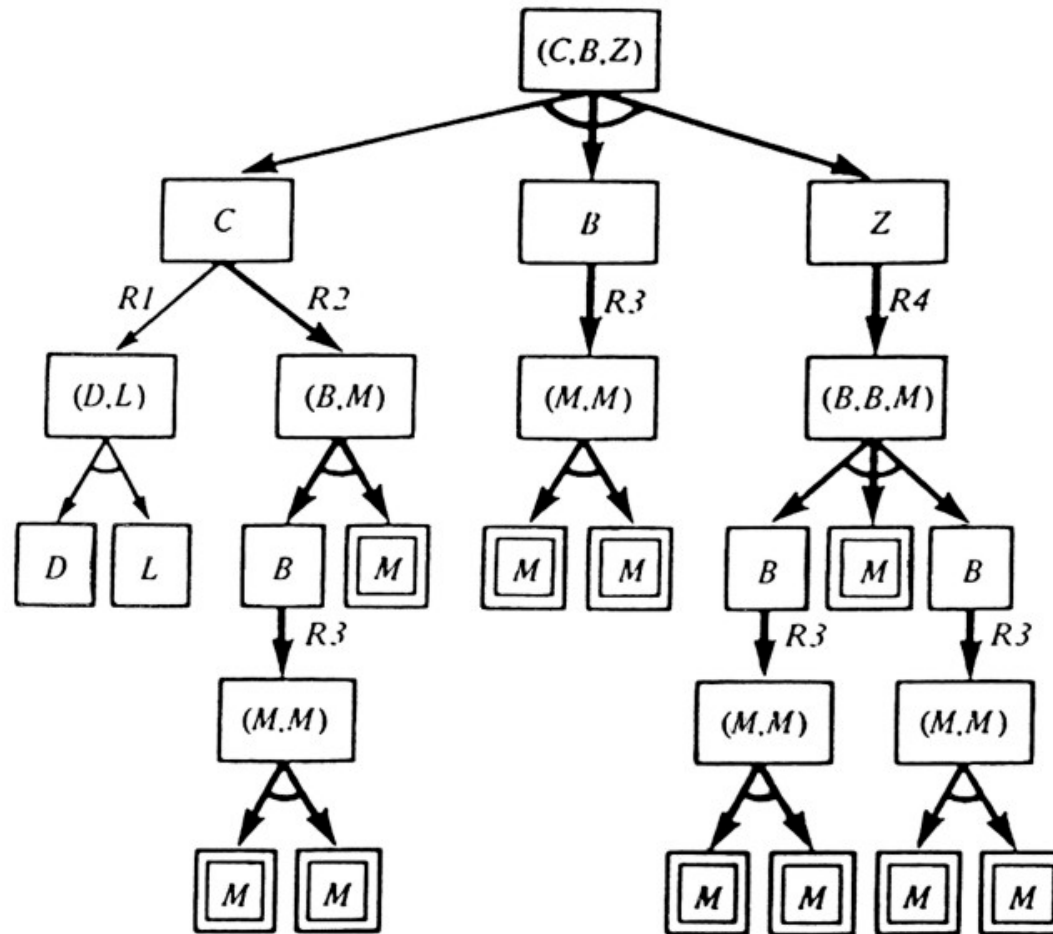
- En el cálculo proposicional, la expresión del grafo Y/O anterior correspondiente sería de la siguiente forma:

$$\mathbf{B \cdot C \rightarrow A; G + D \cdot E \rightarrow B; E + F \rightarrow C}$$

Grafo Y/O

- **Para resolver un grafo Y/O**, cada nodo se resuelve de la siguiente manera:
 - Si es un nodo Y: Resolver todos sus hijos. Combinar la solución y solucionar el nodo. Devolver su solución.
 - Si es un nodo O: Resolver un hijo y ver si devuelve solución. En caso contrario, resolver el siguiente hijo, etc. Cuando ya esté resuelto algún hijo, combinar la solución en el nodo y devolverla.
 - Si es un nodo terminal: Resolver subproblema asociado y devolverla.
- **Mejora:** Para seleccionar el orden de resolución de nodos hijos, se puede utilizar alguna medida de estimación del coste de resolución.

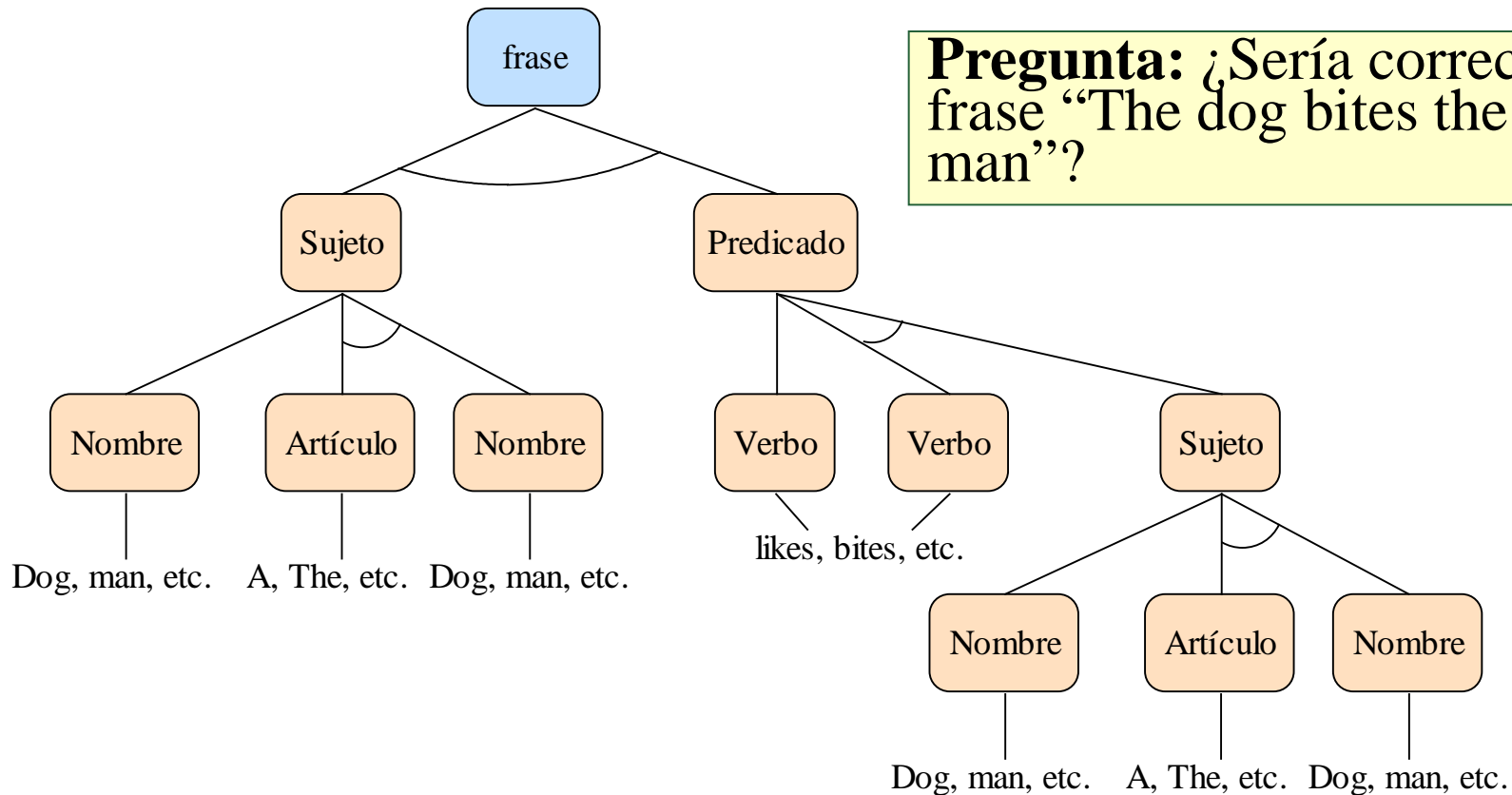
Nueva resolución del problema



Escenario 1: Reconocimiento de frases de lengua inglesa

- Una frase está formada por un sujeto seguido de un predicado.
- El sujeto puede ser un sustantivo o un artículo seguido de un sustantivo.
- El predicado puede ser un verbo, o un verbo seguido de un complemento directo cuya estructura es idéntica a la del sujeto de la frase.

Escenario 1: Reconocimiento de frases de lengua inglesa



Pregunta: ¿Sería correcta la frase "The dog bites the man"?

Escenario 2: Resolución de integrales

- Para simplificar, supongamos que el computador conoce las transformaciones y técnicas de integración, incluidas en una Base de Datos o de Conocimiento.
- Esta técnica es la que implementa el programa MACSYMA, muy utilizado por matemáticos.
- Supongamos que queremos hacer la siguiente integración:

$$\int \frac{x^4}{(1-x^2)^2} dx$$

Escenario 2: Resolución de integrales

