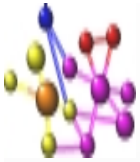
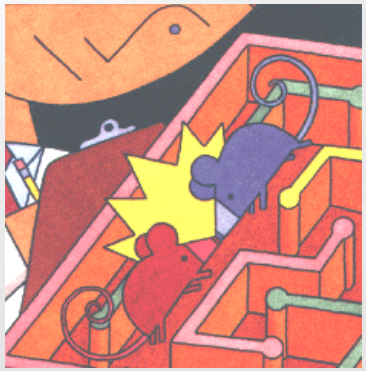
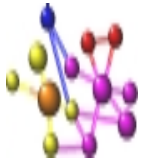


Búsqueda en espacio de estados





Introducción (1)

- Para construir un sistema que resuelva un problema específico, es necesario realizar estas cuatro acciones:
  - Definir el problema con precisión (especificaciones precisas tanto de las situaciones iniciales como sobre las situaciones finales que representan la solución del problema)
  - Analizar el problema, para en función de sus características determinar que técnicas de solución son convenientes y cuales no.
  - Aislar y representar el conocimiento necesario para resolver el problema
  - Elegir la mejor técnica o combinación de técnicas y aplicarla para resolver el problema.

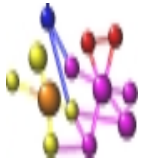
Introducción (2)

- Los problemas se pueden dividir en dos tipos:
  - Aquellos con un método específico de solución.
  - Problemas no totalmente formalizables.
- **Problema bien definido**: Es aquel en que para toda posible solución existe un procedimiento que:
  - Permite verificar automáticamente que se trata de una auténtica solución.
  - Si es una solución, lo verifica en un número finito de pasos.
- Técnicas generales de solución de problemas
  - Conjunto de técnicas de representación y búsqueda que permiten resolver problemas automáticamente.

Aspectos de representación (1)

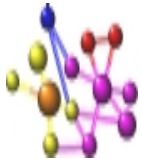
- **Representación**: Conjunto de convenciones sobre la forma de describir un tipo de cosas
- **Descripción**: aprovecha las convenciones de una representación para describir alguna cosa en particular

El hallar la representación apropiada es una parte fundamental de la resolución de un problema. Una vez que un problema se describe mediante una buena representación, el problema está casi resuelto.



Aspectos de representación (2)

- Las **buenas representaciones** hacen explícitos los objetos y las restricciones importantes.
- Agrupan los objetos y las relaciones.
- Suprimen los detalles insignificantes.
- Son transparentes: se entiende lo que dicen.
- Están completos.
- Son concisos: dan la información de forma eficiente.



Aspectos de representación (3)

- Una representación consiste en las siguientes partes fundamentales:
  - Una parte de **léxico** que determina qué símbolos están permitidos en el **vocabulario** de la representación.
  - Una parte **estructural** que describe las restricciones sobre la forma en que los símbolos pueden ordenarse (**sintaxis**).
  - Una parte **operativa** que especifica los **procedimientos de acceso** que permiten crear descripciones, modificarlas y responder a preguntas utilizándolas.
  - Una parte **semántica** que establece una forma de asociar el **significado** con las descripciones.

Búsqueda en espacio de estados

- La búsqueda en espacio de estados es una técnica general de solución de problemas, es un **métodos de generación y prueba**: procedimiento constructivo que:
  - Genera y enumera posibles soluciones de un problema bien definido.
  - Verifica las soluciones generadas.
- La solución de problemas mediante búsqueda en espacio de estados (*EE*) se caracteriza por:
  - Definir formalmente el problema mediante la necesidad de convertir una situación inicial dada (**estado inicial**) en una situación final deseada (**estado final**) usando un conjunto de operaciones permitidas (**operadores**).

Búsqueda en *EE*

- La representación de *EE* emplea las siguientes entidades:
  - **Estados**: estructura de datos que describen la configuración del problema.
  - **Operadores**: conjunto de funciones parciales que permiten transformar un estado en otro. Los operadores tienen dos partes:
    - **Condición**: descripción de los estados a los que se les puede aplicar el operador. Se usa a modo de patrón para determinar los estados en los que es de aplicación.
    - **Acción**: modificación a realizar sobre el estado o descripción estado resultante.

Selección de estados

- Expresividad: capaz de representar cualquier posible configuración del problema
- Concisión: incluir sólo los aspectos importantes para la solución del problema.
- Naturalidad: representación natural del problema.
- Eficiencia computacional: ejecución eficiente del código.

Selección de operadores

- Se han de incluir todos los operadores necesarios para realizar sobre laos estados todas las operaciones que se realizan sobre el mundo real.
- El conjunto de operadores no es único, algunos criterios de selección:
  - Grupo de operadores natural.
  - Número de operadores reducido.
  - Compromiso entre reglas que describen sólo el problema y reglas que describen tanto el problema como algún tipo de conocimiento sobre su solución.
  - Reglas generales.

Formulación de un problema en forma de *EE*

① Definir un espacio de estados que contenga todas las configuraciones posibles de los objetos más relevantes (y quizás algunos imposibles). Es, por supuesto, posible definir este espacio sin tener que hacer una enumeración de todos y cada uno de los estados que contiene.

② Identificar uno o más estados que describan situaciones en las que comience el proceso de resolución del problema. Estos se denominan **estados iniciales**.

③ Especificar uno o más estados que pudieran ser soluciones aceptables del problema. Por alguna propiedad medible del estado, o de la secuencia de estados. Estos estados se denominan **estados finales**, **objetivo** o **meta**.

④ Especificar un conjunto de reglas que describan las acciones (**operadores**) disponibles.

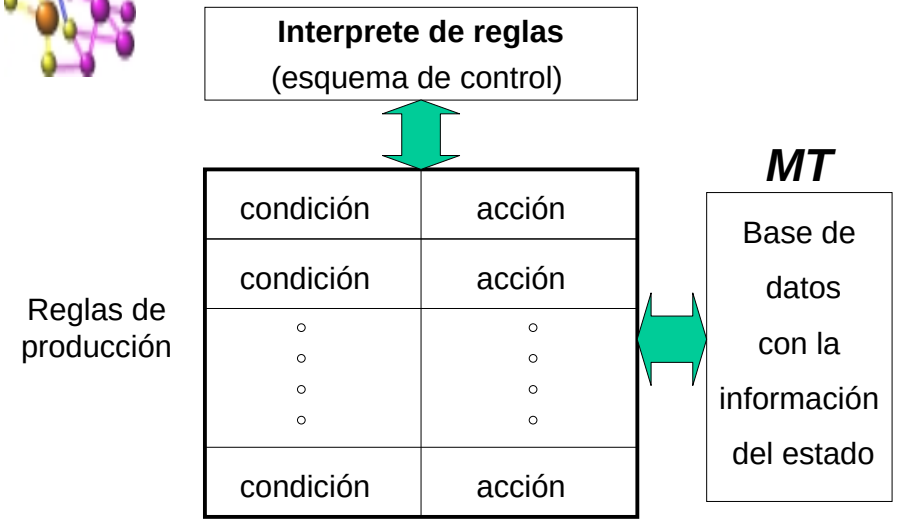
La solución del problema vendrá dada por una secuencia finita de operadores que transforma el estado inicial en un estado final.

Sistema de producción (1)

- Debido a que la búsqueda es el núcleo de muchos procesos inteligentes, es adecuado estructurar los programas de IA de forma que se facilite describir y desarrollar el proceso de búsqueda.
- Los **sistemas de producción** proporcionan tales estructuras.
- Un sistema de producción consta de:
  - memoria de trabajo (estado).
  - conjunto de reglas de producción (operadores).
  - interprete de reglas (control).

Sistema de producción (2)

- **Memoria de trabajo**: estructura de datos usada en un *SP*, contiene la descripción del estado actual del problema.
  - Su contenido depende de la aplicación:
    - Aplicación simple: lista símbolos, matriz numérica,...
    - Aplicación compleja: grafo, cto de ficheros relacionados, ...
- **Reglas de producción**: estructuras de la forma: condición → acción
  - **Condición**: patrón que puede ser satisfecho o no por la memoria de trabajo. Si se satisface, la acción se puede aplicar.
  - **Acción**: modificación de la memoria de trabajo.
- **Interprete de reglas** o **motor de inferencia**: responsable del control:
  - elegir reglas aplicables (*reglas activas*) (orden reglas, *equiparación*, *resolución de conflictos*).
  - aplicar reglas, modificando memoria de trabajo.
  - suspender el proceso si memoria de trabajo satisface alguna condición de terminación.



Sistema de producción (3)

- Interprete básico de un *SP* para realizar búsqueda en *EE*

proc PRODUCCIÓN(*EI*, *F*, *O*)
MT← estado inicial
until *MT* contenga un estado final **do**:
begin
  select algún operador O_i aplicable a *MT*
  MT←resultado de aplicar O_i a *MT*
end

Ejemplo

(a) Si *x* es null, preguntar al usuario por un valor
(b) si *x* no es null y es mayor que 39, imprimir “demasiado grande” y hacer que *x* sea null
(c) si *x* no es null y esta entre 10 y 39, imprimir “X” y restar 10 a *x*
(d) si *x* no es null y es igual a 9, imprimir “IX” y hacer x igual a 0
(e) si *x* no es null y esta entre 5 y 8, imprimir “V” y restar 5 a *x*
(f) si *x* no es null y es igual a 4, imprimir “IV” y restar 1 a *x*
(g) si *x* no es null y esta entre 1 y 3, imprimir “I” y restar 1 a *x*
(h)Si *x* no es null y es igual a 0, imprimir un fin de línea y hacer *x* igual a null

Ejemplo

```
(defun roman2 ()
(let (x)
  (loop
    (cond
     ((null x)(format t "num: ")(setf x (read)))
     ((> x 39)(format t "es grande")(setf x nil))
     ((> x 9)(prin1 'x)(decf x 10))
     ((= x 9)(prin1 'ix)(setf x 0))
     ((> x 4)(prin1 'v)(decf x 5))
     ((= x 4)(prin1 'iv)(setf x 0))
     ((> x 0)(prin1 'i)(decf x 1))
     ((zerop x)(setf x null)(terpri))
     ) ) ) )
```



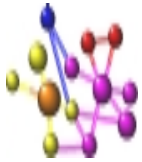
```
(defun roman3 ()
(let (x)
  (loop
    (cond
      ((null x)(format t "num: ")(setf x (read)))
      (t (cond
           ((> x 39)(format t "es grande")(setf x nil))
       (t (cond
           ((> x 4)
            (cond
               ((> x 9)(prin1 'x)(decf x 10))
          (t (cond
              ((= x 9)(prin1 'ix)(setf x 0))
              (t (prin1 'v)(decf x 5)) )) ))
                (t (cond
                 ((= x 4) (print 'iv)(setf x 0))
            (t (cond
                 ((> x 0)(prin1 'i)(decf x 1))
                (t (terpri)(setf x nil)) )) )) )) ))
 ) ) )
```

Control búsqueda en *EE*

- Estrategia de control: hace referencia a:
  - el criterio de selección de reglas
  - el registro de la secuencia de reglas, y
  - las modificaciones de la memoria de trabajo.
- Criterios de clasificación de estrategias de control:
  - Dirección búsqueda: adelante/atrás.
  - Información utilizada: no informada/informada.
  - Objetivo búsqueda: satisfacción/optimización.
  - Reconsideración del efecto de reglas: irrevocable/tentativo.

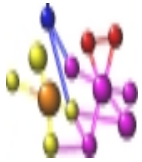
Dirección de búsqueda (1)

- Búsqueda hacia delante (encadenamiento hacia delante o búsqueda guiada por datos)
  - Partimos del estado inicial.
  - Aplicamos reglas hasta alcanzar estado final.
- La dirección de búsqueda se puede invertir sin más que:
  - intercambiar estado inicial/final
  - interpretación inversa de las reglas
- Interpretación de reglas: *Ri*: *Xi*→*Yi*
  - **Interpretación directa**: si estado A satisface *Xi*, realizar acción *Yi* para obtener estado *B* (*D*-reglas).
  - **Interpretación inversa**: si a *B* se le puede aplicar la acción inversa de *Yi*, obtener estado *A* (*I*-reglas).



Dirección de búsqueda (2)

- Búsqueda hacia atrás (encadenamiento hacia atrás o búsqueda guiada por metas)
  - Partimos del estado final.
  - Aplicamos las reglas interpretación inversa hasta alcanzar el estado inicial.
- Búsqueda bidireccional: Se explora simultáneamente en los dos sentidos. Interesante cuando hay garantías de que las fronteras de exploración se encuentran.
- Diferencias entre encadenamiento adelante/atrás: Formalmente ninguna



Criterios de selección: adelante/atrás

- **Número de estados iniciales/finales**: ir del conjunto con menos elementos hacia el que tiene más.
- **Disponibilidad datos y metas**:
  - hacia delante: casi todo lo que conocemos sobre el problema está dado en forma de datos.
  - hacia atrás: se puede formular con facilidad hipótesis que solucionen el problema.
- **Factor de ramificación medio**: número medio de estados accesibles desde cada estado, es preferible ir en la dirección en que el factor de ramificación sea menor.
- **Necesidad de justificar pasos hacia la solución**: Si el sistema ha de justificar su actividad, es importante ir en la dirección que el usuario final considere más importante.

Información utilizada (1)

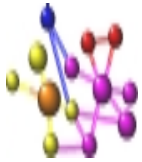
- Sin información, búsqueda exhaustiva o búsqueda ciega: No utiliza ningún criterio para seleccionar una regla de entre las aplicables.
  - Inconveniente: explosión combinatoria, el número de estados a considerar crece de modo exponencial o factorial con el número de reglas aplicadas.
- Con información, búsqueda heurística o búsqueda guiada: Usan conocimiento sobre la naturaleza del problema para seleccionar la regla. El conocimiento se refleja en heurísticas.
  - Heurística: cualquier método que reduzca el esfuerzo de exploración del *EE*, aunque no garantice encontrar solución.
  - Buena heurística: garantiza solución+máxima reducción del esfuerzo de exploración.

Información utilizada (2)

- Métodos de búsqueda heurística:
  1. Métodos de búsqueda ordenada: Deciden que estado se va a explorar primero. Utilizan *funciones de evaluación heurística*:

     $f: U \rightarrow IR$

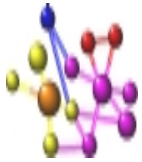
     con la condición de que f tenga un mínimo en los estados del EE, se explora el estado con menor valor de f.
  2. Métodos de poda: Ciertos estados se descartan y nunca se examinan sus sucesores (hay garantías de que no conducen a la solución)



Información utilizada (3)

- Podemos dividir el costo de computación de un SP en dos partes:
  - Costo de control: implementación búsqueda + selección reglas.
  - Costo de aplicación: número de reglas aplicadas y estados generados.
- Influencia de la información en ambas partes:
  - A mayor información, mayor costo control y menor costo aplicación.
  - A menor información, menor costo control y mayor costo aplicación.

El mínimo costo no corresponde a la máxima información



Información utilizada (4)

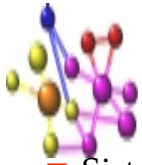
- Ventajas de las heurísticas:
  1. **Si son adecuadas pueden reducir la explosión combinatoria que se produce en muchos problemas.**
  2. Normalmente no se necesita una solución óptima, con frecuencia una buena aproximación es adecuada.
  3. Si bien las aproximaciones que se logran con heurísticas no son muy buenas en los perores casos, estos perores cosos raramente ocurren en el mundo real.
  4. Intentar comprender porque funciona una heurística, o porque no lo hace, normalmente sirve para profundizar en la comprensión del problema

Tipos de búsquedas heurísticas

- Métodos aproximados
  - Escalada simple (simple hill climbing)
  - Escalada profunda (steepest-ascent hill climbing)
- Métodos exactos
  - Primero el mejor (Best First)
  - Beam Search
  - Algoritmo *A*
  - Algoritmo A^*
- Búsquedas con contrincante
  - Búsqueda Mini-Max
  - Búsqueda Alfa-Beta

Reconsideración efecto reglas (1)

- Control irrevocable: una regla aplicable que se selecciona es aplicada irrevocablemente, sin que sea posible una reconsideración posterior.
  - Ventaja: sencillo de implementar, poca carga computacional
  - Inconveniente: la aplicación de una regla “no adecuada” puede impedir alcanzar la solución.
- Interesante en los problemas donde la aplicación de una regla no impide alcanzar la solución (sistemas de producción conmutativos).



Reconsideración efecto reglas (2)

- Sistemas de producción conmutativos: aquellos en los que:
  - La aplicación de una regla R_1, no evita la posterior aplicación de una regla R_2, que fuese aplicable cuando se aplico R_1.
  - Si la aplicación de una secuencia de reglas transforma el estado *X* en el estado *Y*, cualquier permutación de reglas de la secuencia también transforma *X* en *Y*.
- Interés: se puede aplicar régimen de control irrevocable.
- No existe relación entre clases de problemas/tipos de *SP*:
  - Algunos problemas dan lugar, de forma natural a *SP* conmutativos (cuando la solución es añadiendo datos y no modificando).
  - Cualquier problema resuelto por un *SP*, se puede resolver con un *SP* conmutativo.

Reconsideración efecto reglas (3)

- Control tentativo: se selecciona una regla aplicable y se aplica, pero tomando las medida precisas para poder retornar a ese punto del proceso y aplicar entonces otra regla en la misma forma.
  - Ventaja: la aplicación de una regla no impide nunca alcanza la solución.
  - Inconveniente: implementación más costosa, mayor carga computacional.
- Su implementación requiere, al menos, conservar el contenido de la *MT* antes de aplicar una regla.
- Tipos de régimen tentativo:
  - Búsqueda retroactiva: mantiene secuencia de estados recorridos.
  - Búsqueda en grafos: mantienen un grafo para registra los efectos de la aplicación de varias secuencias de reglas.