Métodos de Inteligencia Artificial

L. Enrique Sucar (INAOE)

esucar@inaoep.mx

ccc.inaoep.mx/esucar

Tecnologías de Información

UPAEP

Representaciones híbridas

- Introducción
- Híbridos internos
- Híbridos externos
- Sistemas de pizarrón
- Sistemas de capas

Introducción

Las diferentes formas de representar conocimiento no son mutuamente exclusivas.

Un esquema híbrido es una combinación de diversas formas de representación de conocimiento para resolver un problema.

Hay dos formas básicas de combinar diversas representaciones: **Externa** e **Interna**.

Híbridos “Externos”

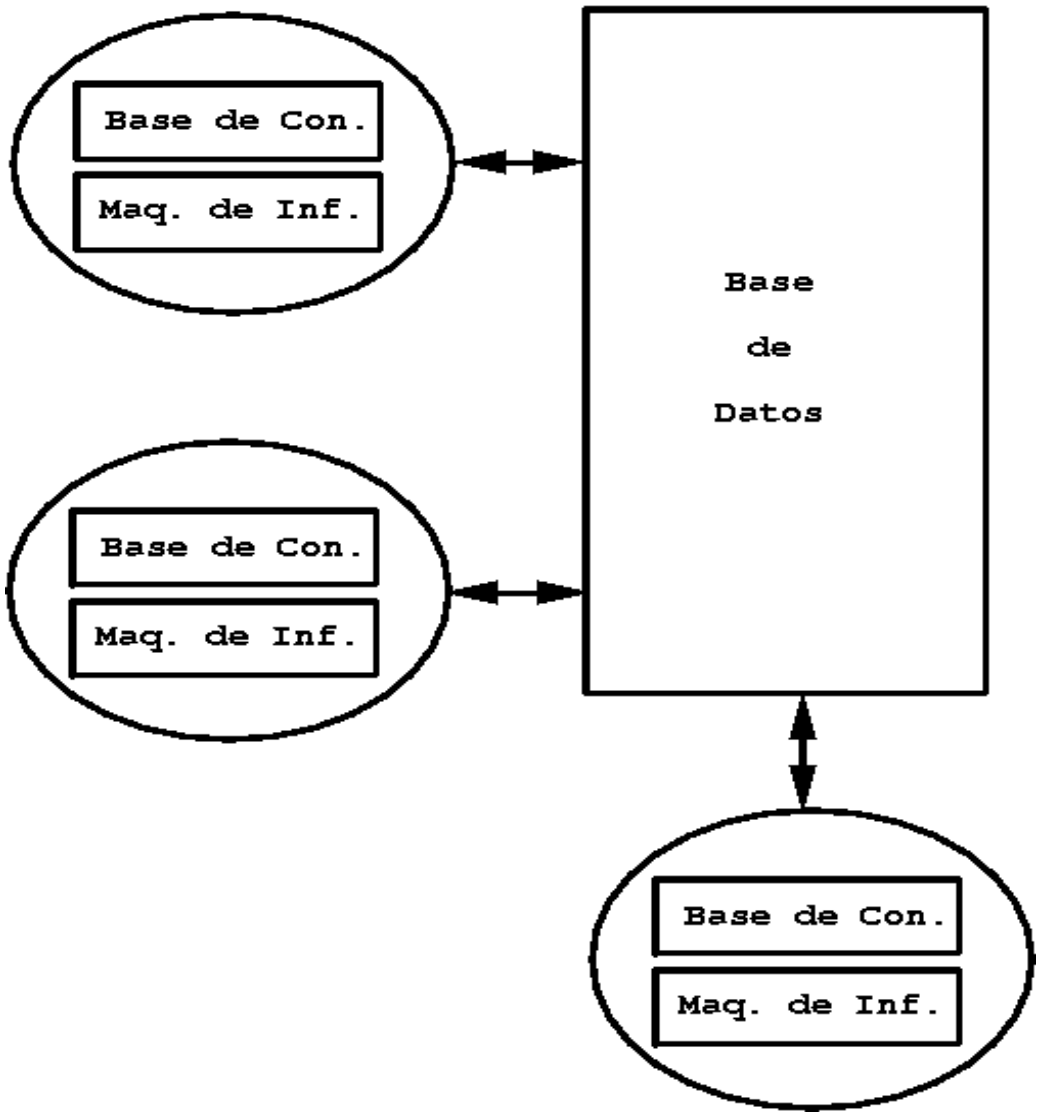
En este esquema dos o más módulos con diferentes formas de representación interactúan entre sí.

Cada módulo tiene una sola forma de representación y se combina con los otros módulos mediante variables de entrada/salida o mediante una estructura de datos común (Base de Datos).

En principio cada subsistema tiene la forma de representación más adecuada para resolver una parte del problema, y se combina con las demás para solucionar un problema mayor.

Este esquema da origen al sistema de *pizarrón*; y, al hacerse en forma distribuida, a los sistemas *multi-agentes*.

Híbridos "Externos"



Híbridos “Internos”

En este tipo de sistemas se combinan varias formas de representación que interactúan para resolver cierto problema.

Con esto se aprovechan diversas propiedades de las formas de representación que complementan sus capacidades.

Por ejemplo, se combinan las reglas con prototipos aprovechando las abstracciones de marcos dentro de reglas.

También se pueden combinar marcos y redes semánticas formando redes de prototipos, etc.

Híbridos “Internos”

Dos ejemplos de este tipo de esquemas son: ***Centaur***, que combina marcos y reglas para diagnóstico médico; y ***Nexpert***, una herramienta que involucra reglas y objetos para el desarrollo de sistemas expertos.

Centaur: Marcos y Reglas

La idea básica es la de asociar reglas a marcos. Es decir, que una estructura tipo marco provee un contexto explícito en el que actúan ciertas reglas.

Las reglas se ligan a un atributo de un marco, y se ven simplemente como un "slot" adicional del prototipo correspondiente.

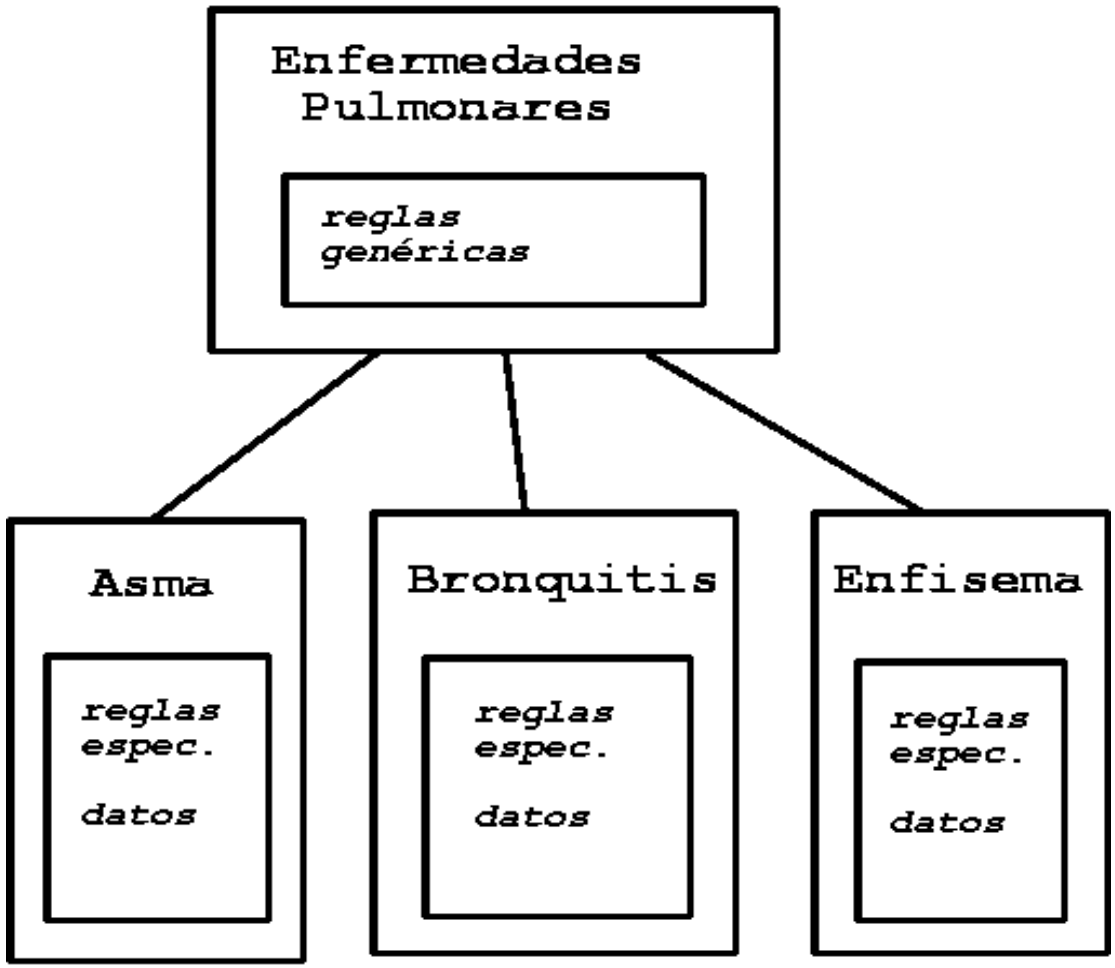
De esta forma, el marco indica la “situación” o contexto en que aplica la regla, evitando los “trucos” que se tienen que hacer en sistemas de producción “puros” para tener un efecto similar.

Centaur

Centaur tiene una clasificación de enfermedades pulmonares que se estructuran en una jerarquía de prototipos (*frames*).

Cada prototipo contiene un número de marcos (sub-prototipos) que incluyen el conocimiento e información referente ese tipo de enfermedad; y asociado a c/u de estos hay una serie de reglas que indican como obtener dicha información.

CENTAUR


Enfermedades Pulmonares
reglas genéricas
Asma
reglas espec.
datos
Bronquitis
reglas espec.
datos
Enfisema
reglas espec.
datos

Centaur

En operación , primero se dan ciertos datos iniciales de la enfermedad.

Estos activan ciertas reglas que llevan a la activación de algunos prototipos.

Se tiene una forma de darles prioridad a los prototipos, y se escoge para su evaluación el de mayor prioridad.

Se obtiene la información referente a ese marco, y se continua el ciclo hasta llegar a cierto nivel de confidencia en los resultados.

Nexpert: Reglas y Objetos

Nexpert es una herramienta (coraza o ***shell***) de propósito general para el desarrollo de sistemas expertos.

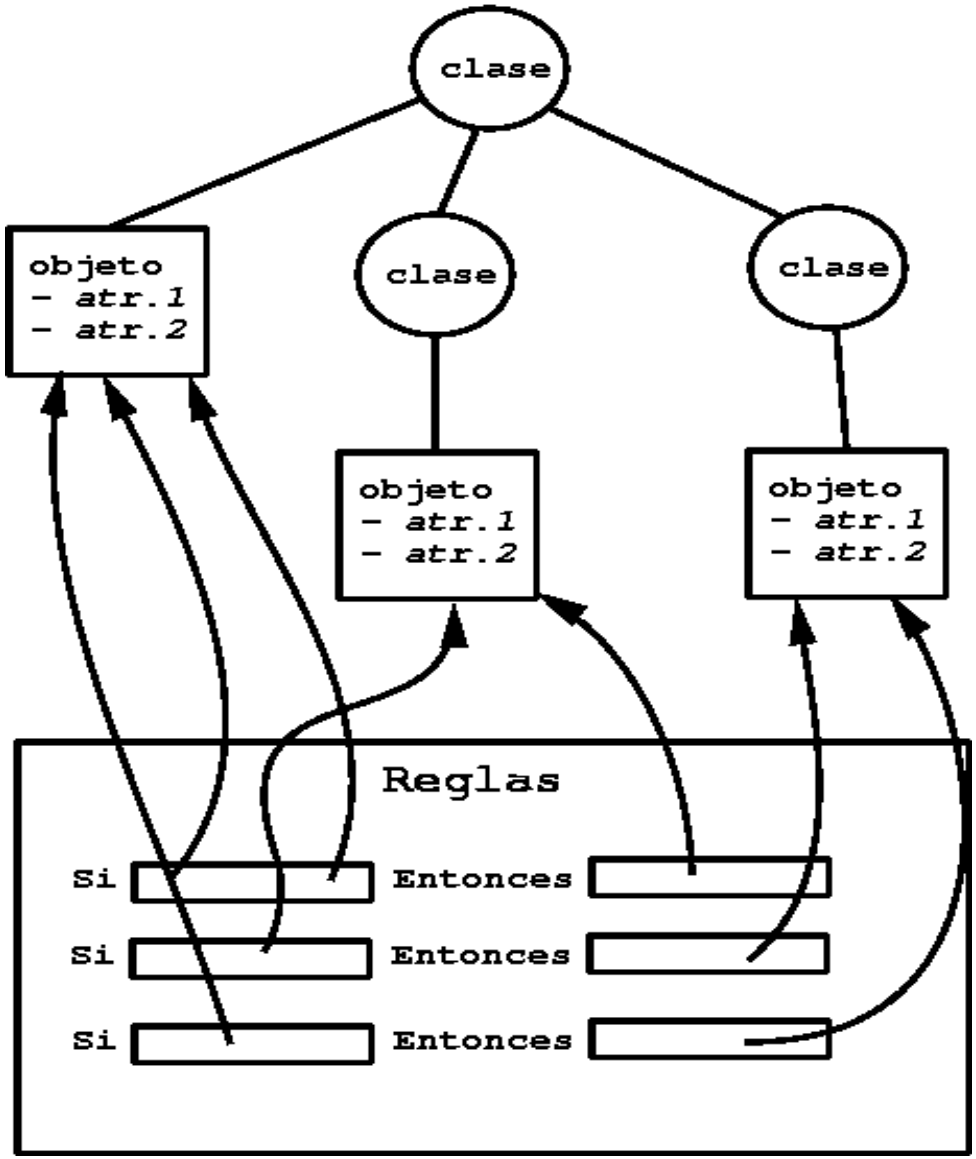
En cierta forma es similar a *Centaur*, ya que los objetos de *Nexpert* se pueden considerar como un sistema de prototipos.

Sin embargo, la forma en que interactúan estas 2 representaciones es diferente, ya que en vez de agrupar reglas dentro de objetos, éstas 2 representaciones se ven como dos dimensiones del conocimiento que interactúan (mediante una intersección) entre sí.

Interacción entre objetos y reglas

- Las reglas operan sobre atributos de objetos.
- Las reglas pueden ser genéricas operando sobre clases de objetos
- Al evaluar reglas se pueden heredar atributos de la jerarquía de clases/objetos y se pueden disparar los métodos para obtener valores (*demons*).

Nexpert Object



Implementación

Una regla puede ser representada por un *frame*:

```
(frame reglaN
        (if (valor: ...)
        (if (valor: ...)
           ...
        (then (valor: ...))
)
```

Implementación

Una regla puede apoyarse en la estructura de los *frames*:

(Regla N

(IF (frame $_1$ propiedad$_{1,i}$ valor$_{1,i}$)

(frame$_2$ propiedad$_{2,j}$ valor$_{2,j}$)

...)

(THEN (frame $_n$ propiedad$_{n,k}$ valor$_{n,k}$)

(frame $_m$ propiedad$_{m,l}$ valor$_{m,l}$)

...))

La regla toma valores de *frames* en sus condiciones y modifica/genera *frames* en sus acciones. También se puede combinar con mecanismos de herencia para obtener valores de *frames* a partir de herencia de sus antecesores.

Shells Híbridos

Algunos sistemas comerciales:

- Nexpert: objetos, reglas
- KEE: frames, reglas, Lisp
- ART: OPS5, TMS
- Knowledge Craft: OPS5, Prolog, CRL

Esquemas de Control

Para resolver problemas complejos, a veces es necesario introducir mecanismos adicionales de control para distribuir tareas.

Existen 3 esquemas principales:

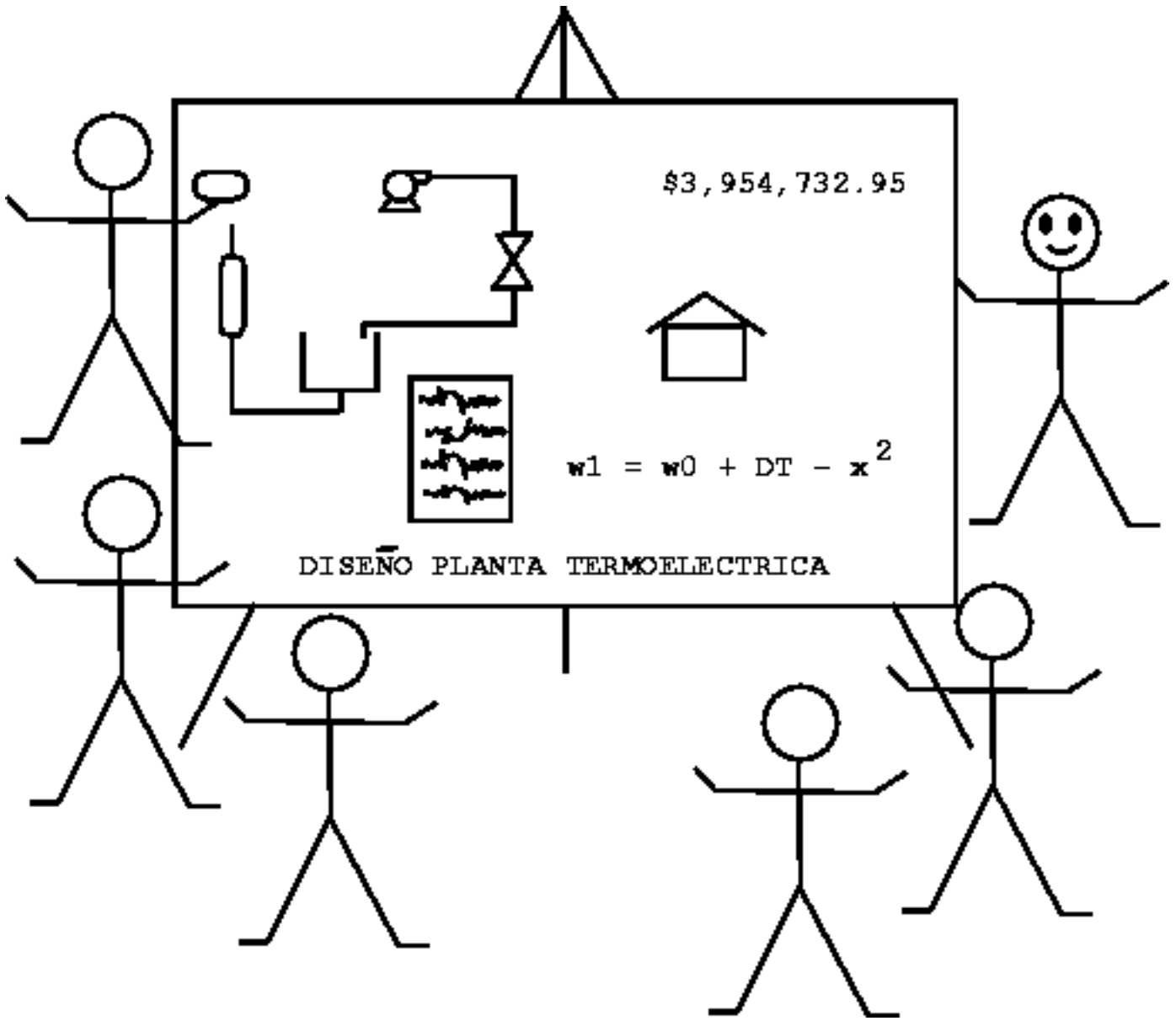
- Arquitectura de Pizarrón
- Sistema de Capas
- Sistemas Multi-agentes

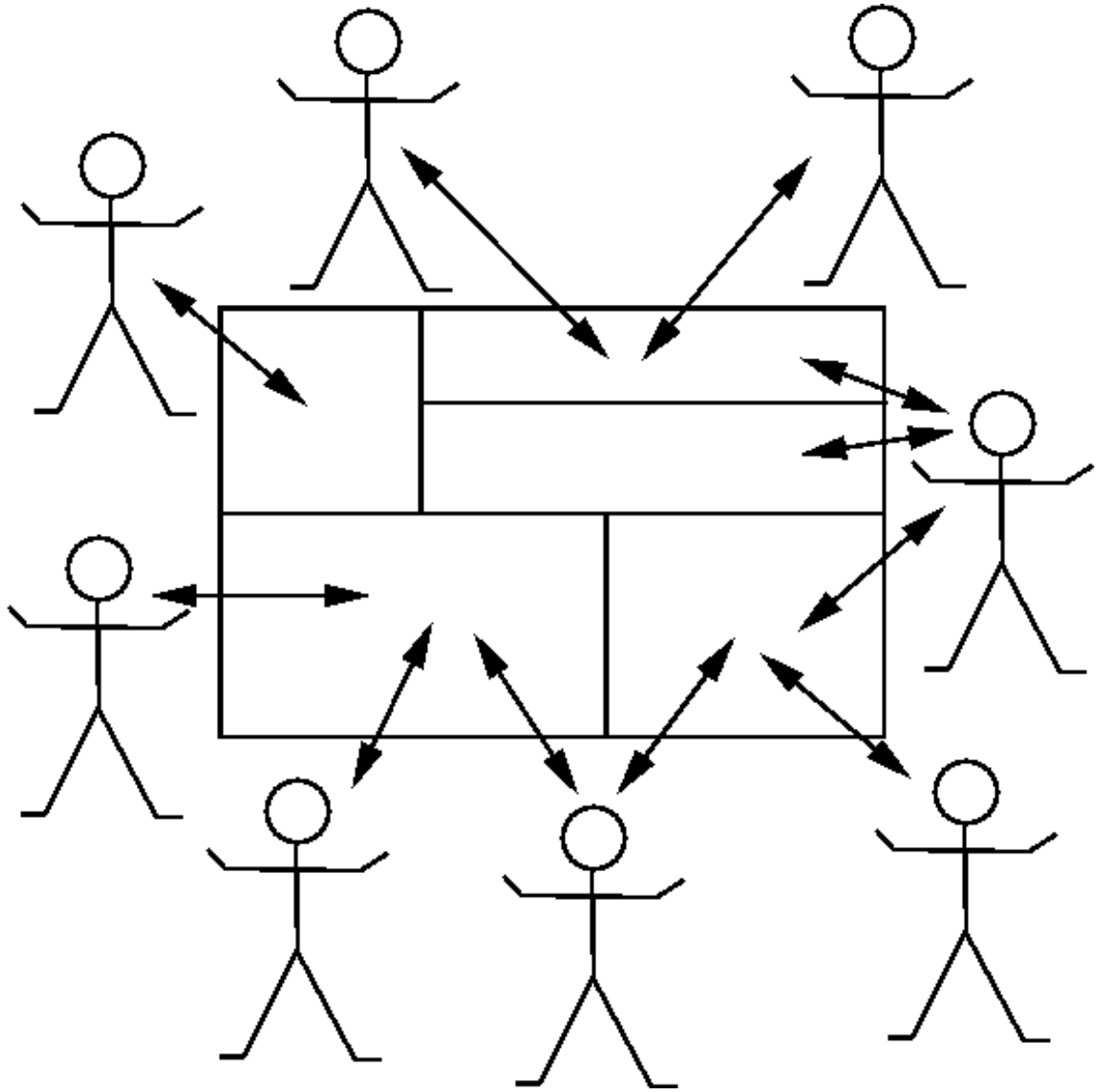
Arquitectura de Pizarrón

Surgió a principios de los 70's para resolver problemas en donde existían varios tipos de “expertos” independientes.

Metáfora:

- Una variedad de expertos cooperando a través de un pizarrón
- Cada experto interviene cuando ve que el estado del pizarrón es tal, que puede contribuir con algo


$3,954,732.95
w1 = w0 + DT - x 2
DISEÑO PLANTA TERMOELECTRICA



Componentes

1. Pizarrón: una estructura de datos multi-dimensional

- Juega el papel de memoria común de comunicación para las KS's
- Almacena toda la información relevante al problema

- La información puede estar almacenada de diferentes maneras: jerarquías, niveles de abstracción, número de hipótesis, etc

- Puede tener información de control

2. Fuentes de Conocimiento (KS's): se pueden ver como sistemas basados en conocimiento

- Juegan el papel de operadores que transforman progresivamente los estados de solución del problema

- **Partes:**
  - **Activación**: condiciones para utilizarse
  - **Evaluación**: estimación (subjetiva) de resultados (recursos, tiempo, beneficios, # de hipótesis)
  - **Acciones**: solución de problemas

Las fuentes de conocimiento pueden ser:

- Genéricas o específicas
- Únicas o redundantes
- Locales o distribuidas
- Homogéneas o híbridas

Las acciones pueden ser:

- Algorítmicas
- Heurísticas

3. Mecanismo de Control (*scheduler*): es el que lleva el razonamiento.

El algoritmo de ejecución "típico" es:

- Ve las nuevas entradas al pizarrón
- Ve cuáles KS's pueden hacer algo
- Construye una agenda de registros de activación de fuentes de conocimiento
- Ordena la agenda con un algoritmo
- Evoca al KS ganador

El mecanismo de control funciona en base a un foco de atención el cual puede estar sobre:

- Las fuentes de conocimiento
- Los objetos del pizarrón
- Una combinación

En algunos sistemas de pizarrón, existen KS's cuyo trabajo es controlar la activación de otros KS's

Al pizarrón le entran mensajes que pueden ser vistos por todos, pero modificados por unos cuantos.

La solución se hace en forma cooperativa.

El proceso termina cuando no hay KS que puedan activarse o cuando se llegó a la solución.

EJEMPLO:

Problema: resolver el siguiente problema de cripto-aritmética:

$$\begin{array}{r} \text{DONALD} \\ +\ \text{GERALD} \\ \hline \text{ROBERT} \end{array}$$

$D = 5$

FdeC 1: dados *c*, *X* y *Y*, calcula S y el nuevo *carry* c'.

$$\begin{array}{r} c \\ X \\ +\ Y \\ \hline S \end{array}$$

$S = c + X + Y$

$c' = 1$ if $S > 9$, else $c' = 0$

FdeC 2: calcula el valor de *Y* o del nuevo *carry*

$$\begin{array}{r} X \\ +\ Y \\ \hline X \end{array}$$

Y = { 0, 9 }

Si Y = 9, entonces c' = 1

FdeC 3: acota el valor de *Y* o de *c*.

$$\begin{array}{r} c \\ X \\ +\ X \\ \hline Y \end{array}$$

Si *Y*= IMPAR, entonces *c* = 1

Si *c* = 1, entonces *Y* = IMPAR

FdeC 4: Dados *c* y *Y*, calcular *X*

$$\begin{array}{r} c \\ X \\ +\ X \\ \hline Y \end{array}$$

$$X = \frac{Y - c}{2} \quad \text{ó} \quad X = \frac{Y - c + 10}{2}$$

Si aparte conozco *c'*

$$X = \frac{Y - c + 10 * c'}{2}$$

FdeC 5: acotar el valor de *S* o de *Y*

$$\begin{array}{r} c \\ X \\ +\ Y \\ \hline S \end{array}$$

Si $c' = 0$, entonces $S > X$ y $S > Y$

Si $c' = 0$ y conozco c y S,
entonces $Y = S - X - c$

FdeC 6: realiza toda la actualización de valores.

- Si una letra es IMPAR sus posibles valores son {1,3,5,7,9}
- Elimina de la lista de posibles valores de las variables los valores ya asignados
- Asigna a una letra un valor si es el único valor posible
- Elimina valores que violen restricciones (v.g., mayor-que)

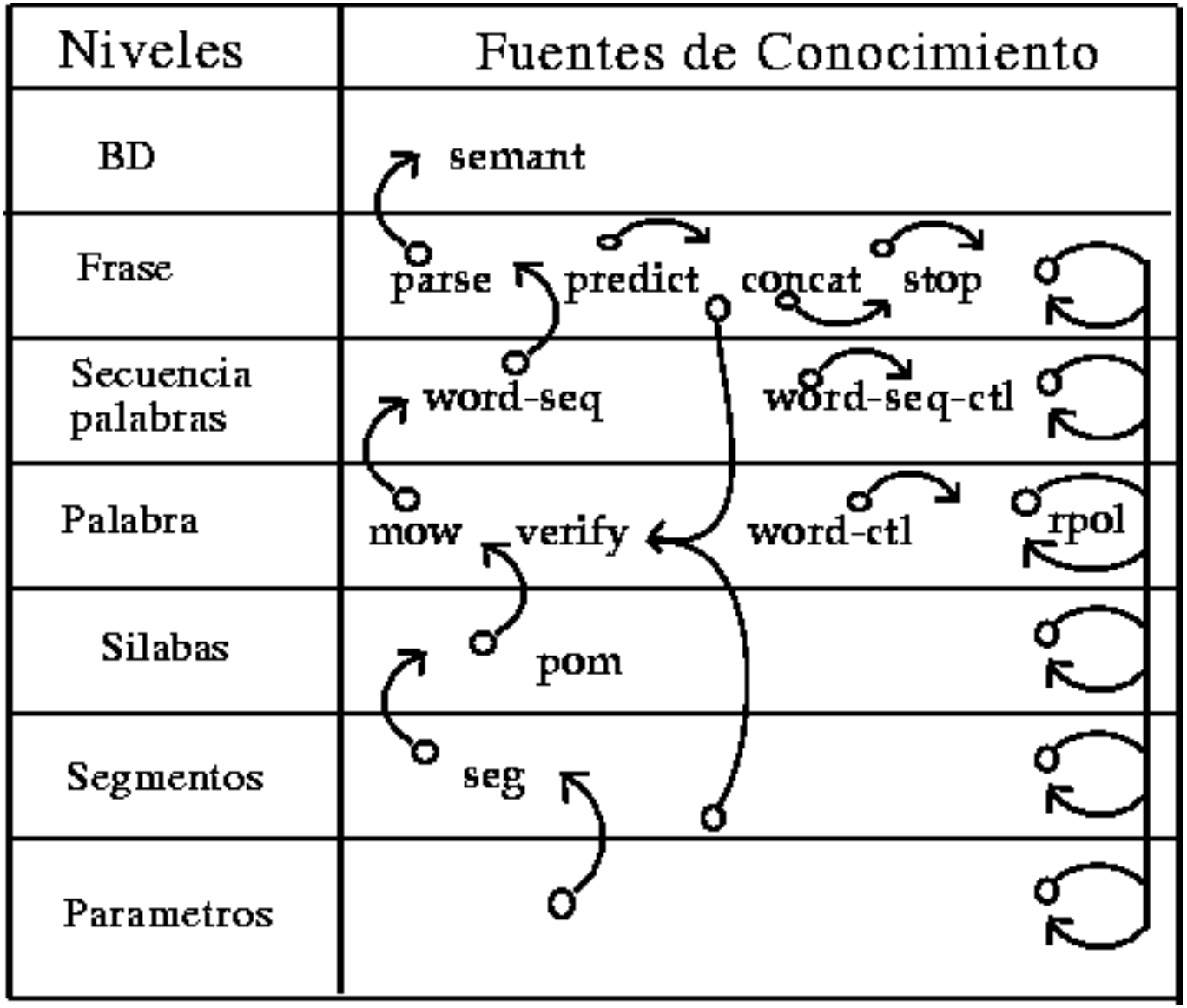
FdeC 7: búsqueda exhaustiva

Hearsay-II

El primer sistema de pizarrón, construido para entender voz.

Resultado de un concurso de ARPA (71): crear un sistema para 1976 que:

- aceptara voz continua
- a través de un buen micrófono
- en una sala silenciosa
- 1,000 palabras
- tiempo real


Niveles
Fuentes de Conocimiento
BD
semant
Frase
parse
predict
concat
stop
Secuencia palabras
word-seq
word-seq-ctl
Palabra
mow
verify
word-ctl
rpol
Silabas
pom
Segmentos
seg
Parametros

Arquitectura de Capas

- Extender la idea de las agendas a diferentes niveles con un solucionador a cada nivel.

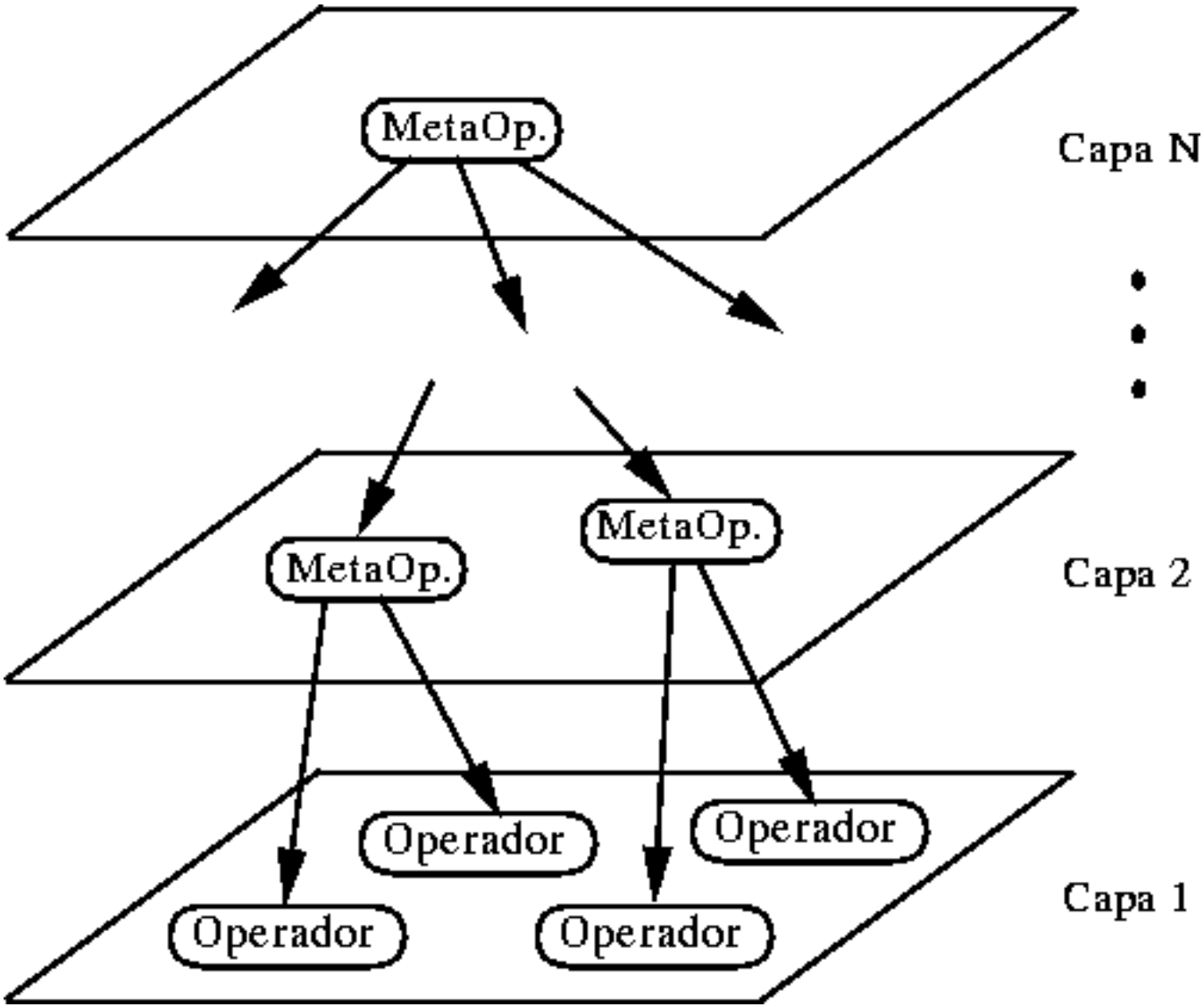
- Organizar los operadores y la información en capas (*layers*)

- Las capas superiores tienen *meta*-operadores que actúan sobre los operadores de la capa inmediata inferior.

- La capa inferior tiene operadores que actúan directamente sobre el estado de solución del problema

El control se implementa capa por capa, siendo la responsabilidad de una capa controlar la ejecución de los operadores (o *meta*-operadores) de la capa inmediata inferior.

La comunicación entre capas se realiza por medio de mensajes.


MetaOp.
Capa N
MetaOp.
MetaOp.
Capa 2
Operador
Operador
Operador
Operador
Capa 1

MOLGEN (Stefik, '81)

Sistema que planea la realización de experimentos en genética molecular.

Utiliza restricciones para reducir la búsqueda e incorpora algoritmos para *formular*, *propagar* y *satisfacer* restricciones.

Las restricciones sirven para:

1. Limitar posibles valores
2. Forzar ciertos valores
3. Comunicación entre subproblemas

MOLGEN tiene 3 capas (cada una con operadores y objetos).

1. **Espacio de laboratorio:**
   Tiene conocimiento de objetos y operaciones de un laboratorio genético

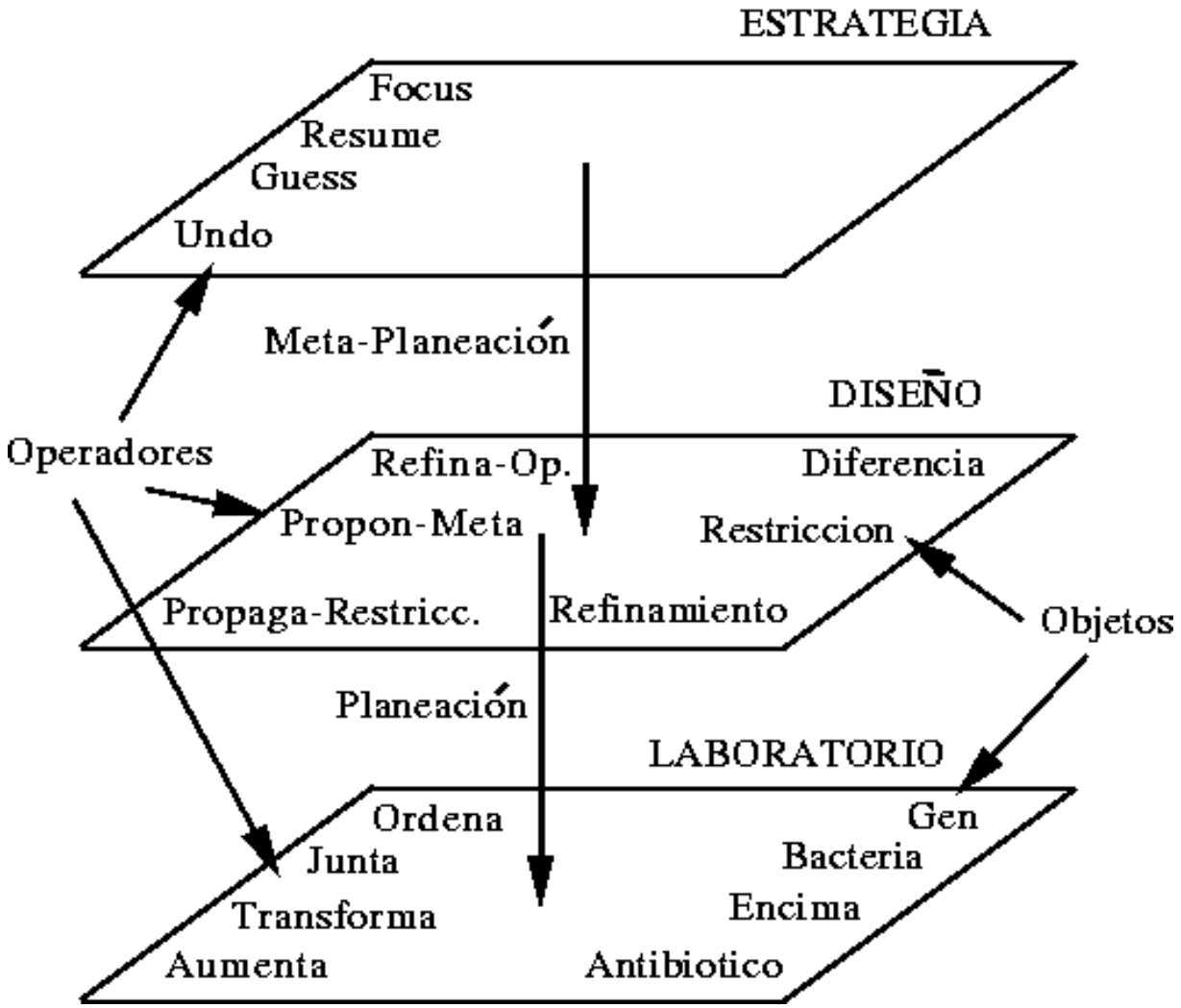
2. Espacio de diseño:

Conocimiento acerca del diseño de planes

3. Espacio de estrategia:

Sigue heurísticas y una estrategia del menor compromiso (*least-commitment*) genera meta-planes

MOLGEN


ESTRATEGIA
Focus
Resume
Guess
Undo
Meta-Planeación
DISEÑO
Operadores
Refina-Op.
Diferencia
Propon-Meta
Restriccion
Propaga-Restricc.
Refinamiento
Objetos
Planeación
LABORATORIO
Ordena
Gen
Junta
Bacteria
Transforma
Encima
Aumenta
Antibiotico

Tarea

- Leer sobre sistemas de Pizarrón (ver ligas en la página)